Statistical Machine Learning HW2

1. (a) $H(X,Y) = H(Y|X) + H(X)$

$H(X,Y) - H(X|Y) - H(Y|X)$

$= H(Y|X) + H(X) - H(X|Y) - H(Y|X)$

$= H(X) - H(X|Y) = IG(X;Y)$

(b) (i) Initial Entropy of Usage is

$n = 15$
$L = 7$
$M = 5$
$H = 3$

$H(S) = -P(Low) \cdot \log_2(P(Low)) - P(Medium) \cdot \log_2(P(Med))$
$- P(High) \cdot \log_2(P(High))$

$= -(7/15) \cdot \log_2(7/15) - (5/15) \cdot \log_2(5/15) - (3/15) \cdot \log_2(3/15)$

$= 1.5058$

(ii) I want to choose the attribute which yields the maximum information gain

First Attribute - Income

categorical values -

	Low			Medium			High		
	5			6			4		
	L	M	H	L	M	H	L	M	H
	5	0	0	2	4	0	0	1	3

$H(Income: Low) = -(5/5) \cdot \log_2(5/5) - 0 - 0 = 0$

$H(Income = Medium) = -(2/6) \cdot \log_2(2/6) - (4/6) \cdot \log_2(4/6) - 0 = 0.9183$

$H(Income = High) = -0 - (1/4) \cdot \log_2(1/4) - (3/4) \cdot \log_2(3/4) = 0.81127$

Average Entropy information for Income.

$H(Usage | Income) = P(Low) \cdot H(Income = Low) + P(Med) \cdot H(Income = Med)$
$+ P(High) \cdot H(Income = High)$

$= \frac{5}{15} \cdot 0 + \frac{6}{15} \cdot 0.9183 + \frac{4}{15} \cdot 0.81127$

$= 0.58365$, Information gain $= 1.5058 - 0.58365 = \boxed{0.92215}$

Second Attribute Age

categorical values -

	Old			Young		
	9			6		
	L	M	H	L	M	H

4 0 2 0 0 1

$H(Age = old) = -(7/9)\cdot\log_2(7/9) - 0 - (2/9)\cdot\log_2(2/9) = 0.7642$

$H(Age = Young) = -0 - (5/6)\cdot\log_2(5/6) - (1/6)\cdot\log_2(1/6) = 0.65$

Average entropy Information for Age

$H(Usage \mid Age) = P(old)\cdot H(Age = old) + P(young)\cdot H(Age = young)$

$= (9/15)\cdot 0.7642 + (6/15)\cdot 0.65$

$= 0.71852$

Information Gain $= H(S) - H(Usage \mid Age)$

$= 1.5058 - 0.71852$

$= \boxed{0.78728}$

Third Attribute - Education

Categorical values - University, College, High School

	University			College			High School		
	6			5			4		
	L	M	H	L	M	H	L	M	H
	3	0	3	0	5	0	4	0	0

$H(Edu = Univ) = -(3/6)\cdot\log_2(3/6) - 0 - (3/6)\cdot\log_2(3/6) = 1$

$H(Edu = College) = -0 - (5/5)\cdot\log_2(5/5) - 0 = 0$

$H(Edu = Highschool) = -(4/4)\cdot\log_2(4/4) - 0 - 0 = 0$

Average Entropy Information for Education

$H(Usage \mid Edu) = P(Univ)\cdot H(Edu = Univ) + P(College)\cdot H(Edu = College)$

$+ P(High)\cdot H(Edu = High)$

$= 6/15 \cdot 1 + 5/15 \cdot 0 + 4/15 \cdot 0$

$= 6/15 = 0.4$

Information Gain $= H(S) - H(Usage \mid Edu)$

$= 1.5058 - 0.4$

$= \boxed{1.1058}$

Fourth Attribute - Marital Status

Categorical values — Single Married

7 8

L M H L M H

2 2 3 5 3 0

$H(\text{Marital} = \text{Single}) = -(2/7)\cdot\log_2(2/7) - (2/7)\cdot\log_2(2/7)$
$- (3/7)\cdot\log_2(3/7) = 1.55665$

$H(\text{Marital} = \text{Married}) = -(5/8)\cdot\log_2(5/8) - (3/8)\cdot\log_2(3/8) - 0 = 0.95443.$

$H(\text{Usage}|\text{Marital}) = P(\text{Single})\,H(\text{Marital} = \text{Single}) + P(\text{Married})\cdot H(\text{Marital} = \text{married})$

$= 7/15 \cdot 1.55665 + 8/15 \cdot 0.95443$

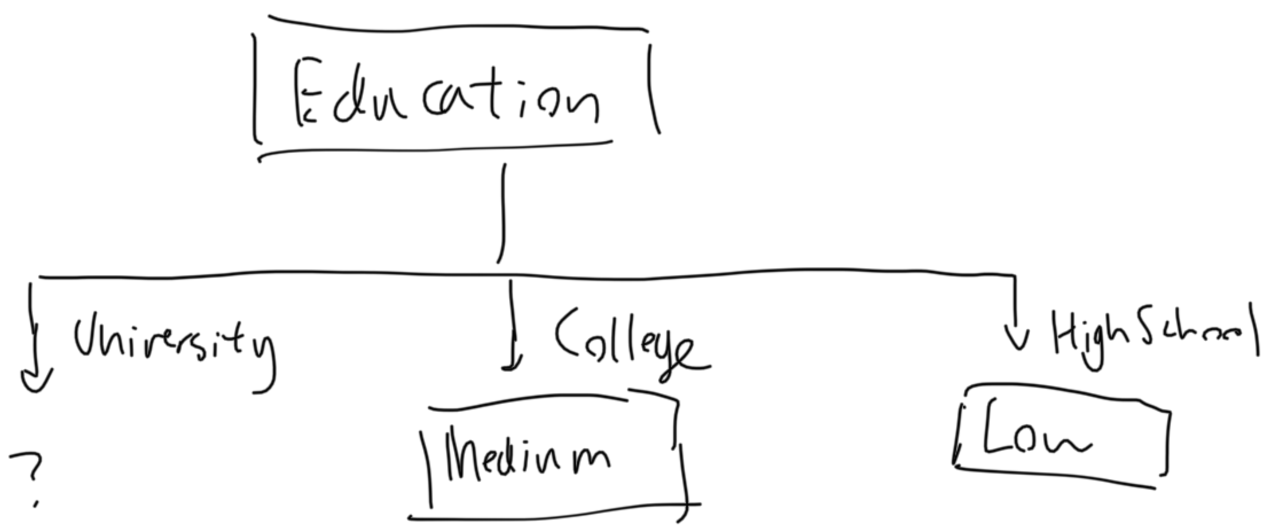
$= 1.23546$

Information Gain $= H(S) - H(\text{Usage}|\text{Marital})$

$= 1.5058 - 1.23546$

$= \boxed{0.27034}$

Here, the attribute with the maximum information gain is Education



Here, when education = college, It's a pure class of medium Usage. when education = High school, It's a pure class of Low usage.. The only thing left is university

Complete entropy of university is.

$H(S) = -(3/6)\cdot\log_2(3/6) - 0 - (3/6)\cdot\log_2(3/6)$

$= 1$

First Attribute. - Income.

Categorical values. - Low Medium High

3 0 3

L M H L M H

3 0 0 0 0 3

$H(\text{Univ}, \text{Income} = \text{Low}) = -(3/3)\cdot\log_2(3/3) - 0 - 0 = 0$

$H(\text{Univ}, \text{Income} = \text{Med}) = -0 - 0 - 0 = 0$

$H(\text{Univ}, \text{Income} = \text{High}) = -0 - 0 - (3/3)\log_2(3/3) = 0$

$I(\text{Univ}, \text{Income}) = 0$

Information Gain $= H(\text{Univ}) - I(\text{Univ}, \text{Income}) = 1$

Second Attribute, Age

categories —	Old			Young		
	5			1		
	L	M	H	L	M	H
	3	0	2	0	0	1

$H(\text{Univ}, \text{age} = \text{old}) = -(3/5)\cdot\log_2(3/5) - (2/5)\cdot\log_2(2/5) - 0 = 0.971$

$H(\text{Univ}, \text{age} = \text{young}) = -0 - 0 - (1/1)\cdot\log_2(1/1) = 0$

$I(\text{Univ}, \text{age}) = 5/6 \cdot 0.971 = 0.80916$

Information Gain $= H(\text{Univ}) - I(\text{Univ}, \text{age}) = 0.19084$

Third Attribute — Marital Status.

categories —	Single			Married		
	3			3		
	L	M	H	L	M	H
	0	0	3	3	0	0

$H(\text{Univ}, \text{marital status} = \text{Single}) = -0 - 0 - (3/3)\cdot\log_2(3/3) = 0$

$H(\text{Univ}, \text{marital status} = \text{married}) = -0 - 0 - (3/3)\cdot\log_2(3/3) = 0$

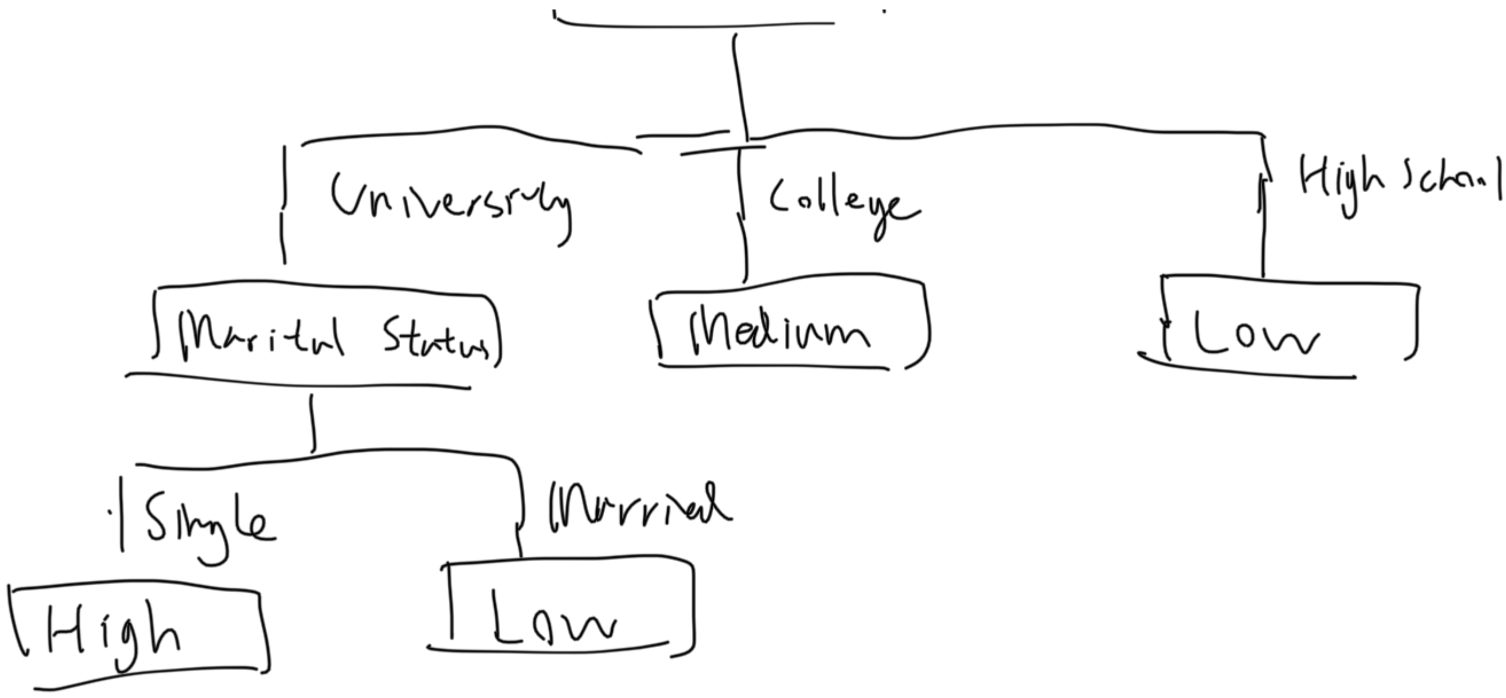
$I(\text{Univ}, \text{marital status}) = 0$

Information Gain $= 1$

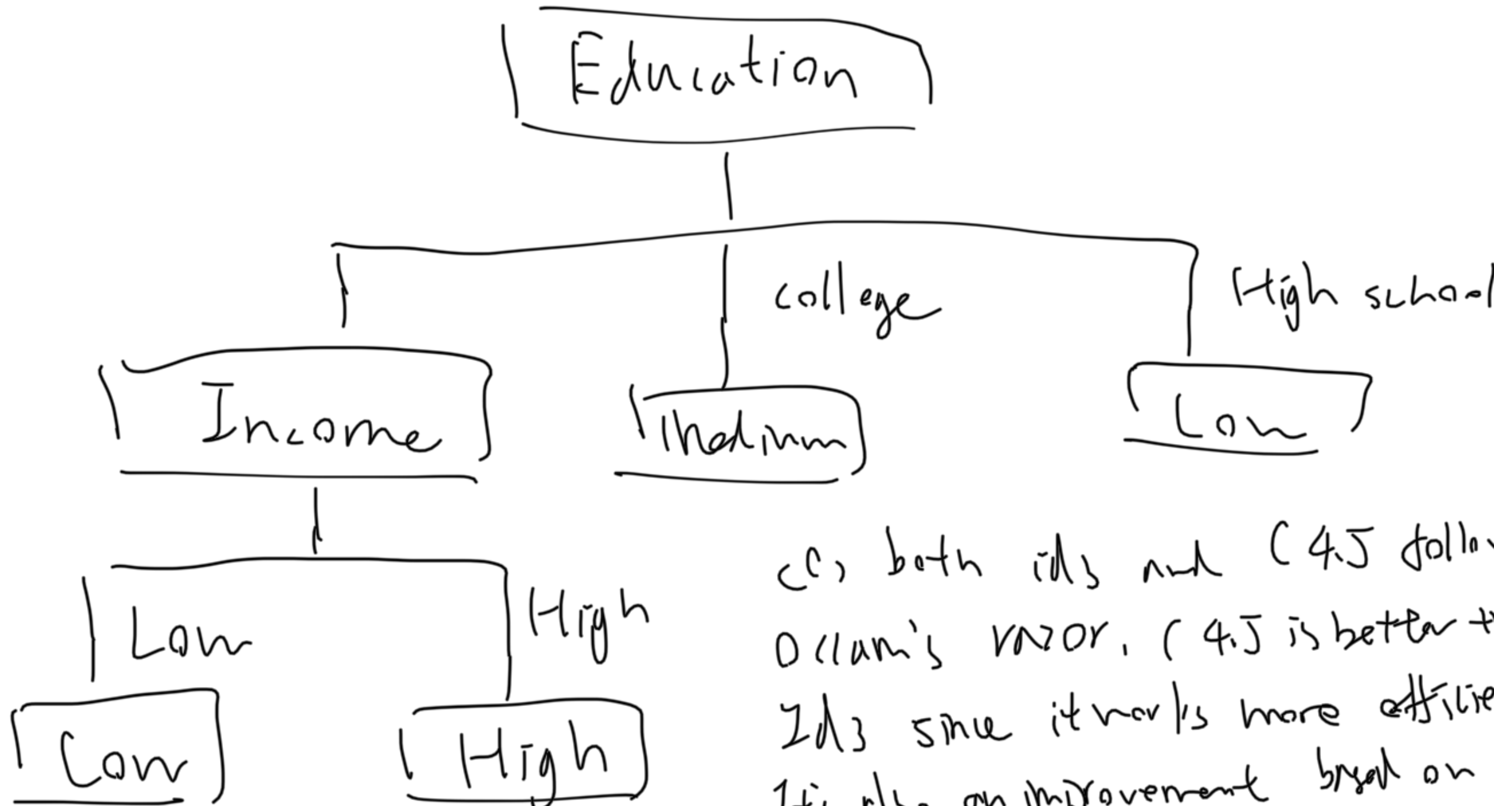
In this case, both marital status and Income could be chosen as the next node.

(iii) choice one — choose marital status as second node

$\boxed{\text{Education}}$

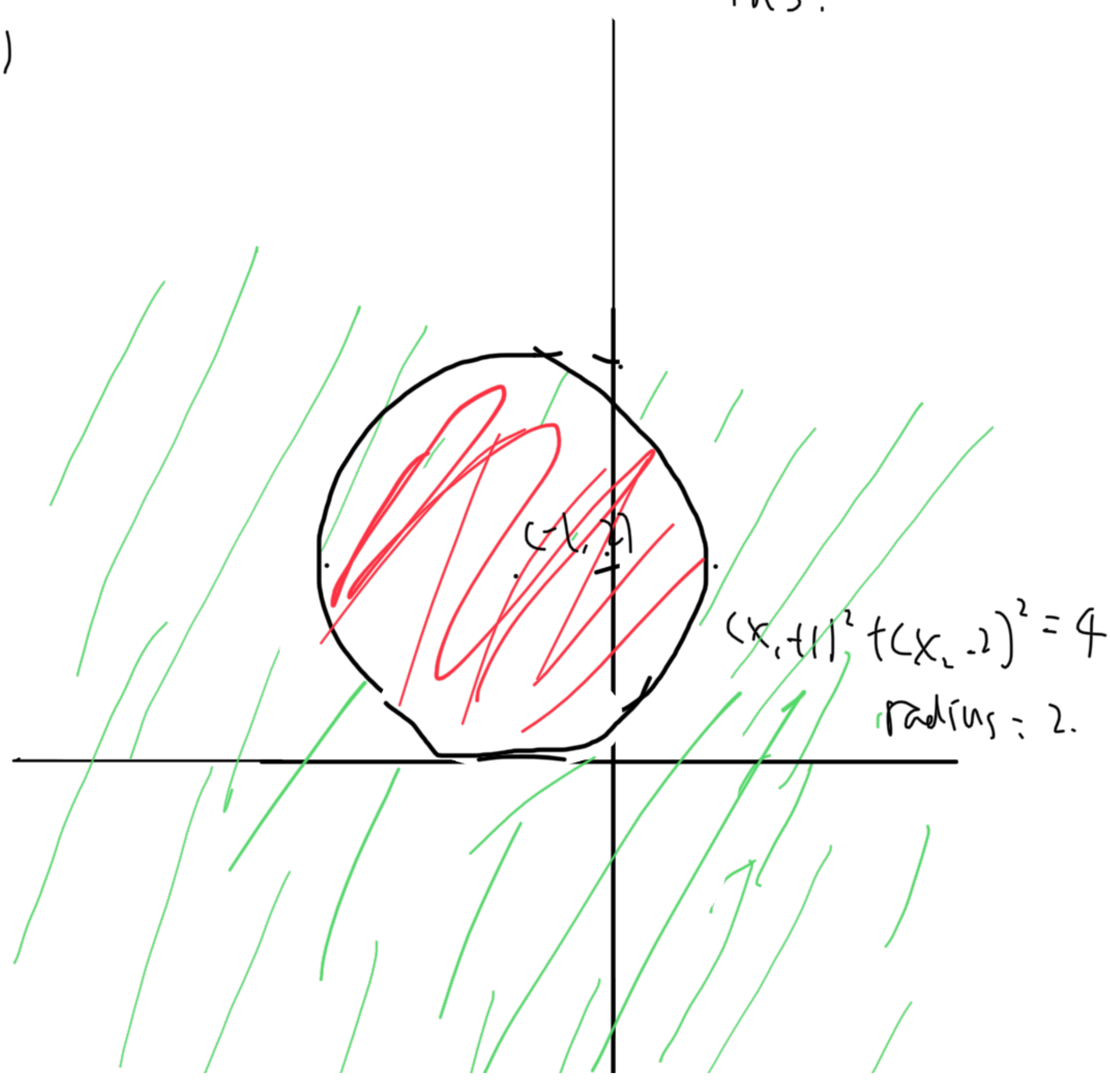


Choice two, Choose income as the second node.



(c) both id3 and C4.5 follow Occam's razor. C4.5 is better than Id3 since it works more efficiently. It's also an improvement based on id3.

2. (a)



(b) the set of points for which $(1+X_1)^2+(2-X_2)^2>4$ is shaded by green.

the set of points for which $(1+X_1)^2+(2-X_2)^2\le 4$ is shaded by red, with points on the circle $(1+X_1^2)^2+(2-X_2)^2=4$ included.

(c) observation $(0,0)$ will fall in green class
observation $(-1,1)$ will fall in red class
observation $(2,2)$ will fall in blue class
observation $(3,8)$ will fall in blue class

(d) $(1+X_1)^2+(2-X_2)^2=4$

$$X_1^2+2X_1+1+X_2^2-4X_2+4=4$$

$$1+2X_1+X_1^2-4X_2+X_2^2=0$$

As we can see that, through transformation, the decision boundary is in form

$$\beta_0+\beta_1X_1+\beta_2X_1^2+\beta_3X_2+\beta_4X_2^2=0$$

This is linear in terms of X_1, X_1^2, X_2, X_2^2, but not linear in terms of only X_1 and X_2.

3. (a)

رط

Graph (a) (b) (c) is linearly separable

(d) is linearly separable with one miss classification.

The change from 1-NN to SVM is illustrated through graph

(c) Higher order polynomial kernels such as quadratic kernel could be applied to figure (d) to make blue and red points linearly separable.

4.(a) The absolute error Loss is

$L = |y - f(x)|$

and the epsilon insensitive loss function will become.

$L_\varepsilon(y, \hat{y}) = |y - \hat{y}|$ since $|y - \hat{y}|$ will always $\geq \varepsilon = 0$

$f(x) = w^T x + b$ $\quad |y - \hat{y}| = |y - f(x)|$

$\hat{y} = w^T x + b$

I would say, when $\varepsilon = 0$, the epsilon insensitive loss function is the same as the absolute error loss.

The ε's function is that, in epsilon insensitive loss function, all the errors $|y - \hat{y}|$ smaller than ε distance of the observed value will be treated as 0.

(b) $J(w) = \frac{1}{n} \sum_{i=1}^{n} L_\varepsilon(y, \hat{y}(x_i)) + \lambda \|w\|_2^2$

$= \frac{1}{n} \sum_{i=1}^{n} (|y - w^T x_i| - \varepsilon) + \lambda \|w\|_2^2$

add slack variable to the objective function.

$$= \frac{1}{n}\sum_{i=1}^{n} (|y - w^T x_i| - \varepsilon) + \lambda \|w\|_2 + \sum_{i=1} \xi_i$$

$$= \frac{1}{n}\sum_{i=1}^{n} (|y - w^T x_i| - \varepsilon + \xi_i) + \lambda \|w\|_2^2$$

since the constrint is

$$L_\varepsilon(y, \hat{y}(x_i)) = \begin{cases} 0 & \text{if } |y - \hat{y}(x_i)| < \varepsilon \\ |y - \hat{y}(x_i)| - \varepsilon & \text{, otherwise.} \end{cases}$$

I would like to add ξ_i to the constraint.

making it $L_\varepsilon(y, \hat{y}(x)) = 0$, $y - \hat{y}(x_i) \le \varepsilon + \xi_i$

i.e. $y - \hat{y}(x_i)$ is always smaller than $\varepsilon + \xi_i$.

making it always give 0 for $L_\varepsilon(y, \hat{y}(x_i))$

now $y - \hat{y}(x_i) \le \varepsilon + \xi_i$

$$-(y - \hat{y}(x_i)) \le \varepsilon + \xi_i$$

$$y - \hat{y}(x_i) \ge -\varepsilon - \xi_i$$

The optimization function becomes

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} \xi_i + \lambda \|w\|_2^2$$

with constraint

$$y - \hat{y}(x_i) \le \varepsilon + \xi_i$$

$$y - \hat{y}(x_i) \ge -\varepsilon - \xi_i$$

and $\xi_i \ge 0$

This is an optimization problem that is differentiable and with linear constraints.

In [1]:
```python
import numpy as np;
import pandas as pd;
import matplotlib.pyplot as plt;
from matplotlib import image;
from sklearn import preprocessing;
from sklearn.tree import DecisionTreeClassifier;
from sklearn import metrics;
from sklearn.model_selection import train_test_split;
from sklearn.tree import plot_tree
from sklearn.neighbors import KNeighborsClassifier;
import seaborn as sns;
from matplotlib.colors import ListedColormap;
from sklearn import svm;
from sklearn.naive_bayes import GaussianNB;
from sklearn.ensemble import AdaBoostClassifier;
from sklearn.preprocessing import StandardScaler;
```

In [2]:
```python
df = pd.read_csv("hw2_Q5.txt", names = ["sepal_length", "sepal_width", "petal_length",
```

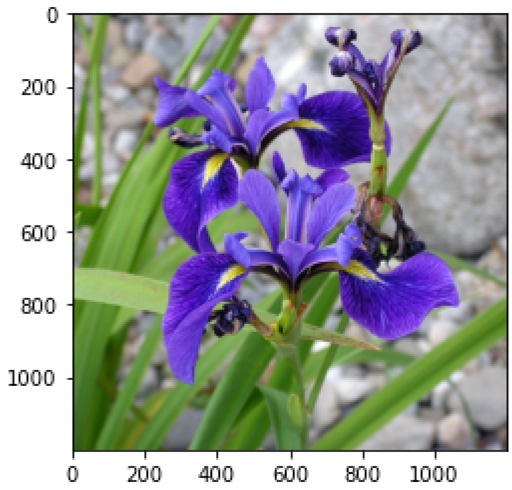
Q5-a

In [3]:
```python
Setosa = image.imread("Setosa.jpg")
print("Image for Setosa")
plt.imshow(Setosa)
plt.show()
```

```
Image for Setosa
```

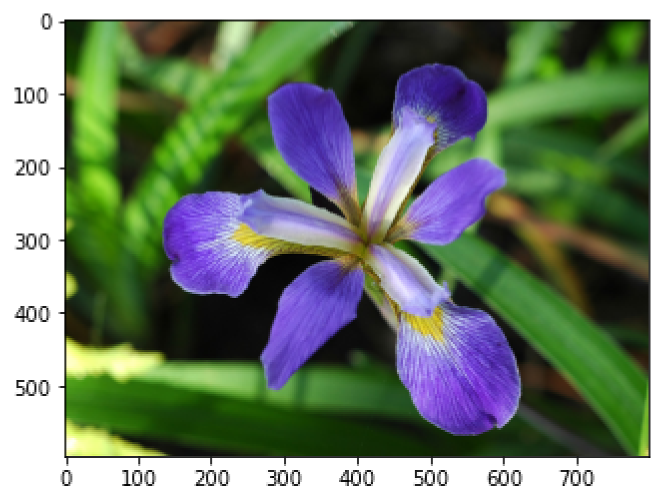
In [4]:
```python
Versicolor = image.imread("Versicolor.jpg")
print("Image for Versicolor")
plt.imshow(Versicolor)
plt.show()
```

```
Image for Versicolor
```



In [5]:
```
Virginica = image.imread("Virginica.jpg")
print("Image for Virginica")
plt.imshow(Virginica)
plt.show()
```

```
Image for Virginica
```



Q5-b

In [6]:
```
#X1 and Y
print(np.corrcoef(df.sepal_length, df.Y))
```

```
[[1.         0.78256123]
 [0.78256123 1.        ]]
```

In [7]:
```
#X2 and Y
print(np.corrcoef(df.sepal_width, df.Y))
```

```
[[ 1.        -0.4194462]
 [-0.4194462  1.       ]]
```

In [8]:
```
#X3 and Y
print(np.corrcoef(df.petal_length, df.Y))
```

```
[[1.         0.94904254]
 [0.94904254 1.        ]]
```

In [9]:

```
#X4 and Y
print(np.corrcoef(df.petal_width, df.Y))
```

```
[[1.         0.95646382]
 [0.95646382 1.        ]]
```

I would discard X2, which is the sepal width first, since it possesses the lowest correlation coefficient with Y among the fou features

Q5-C

In [10]:

```
d = preprocessing.normalize(df)
scaled_df = pd.DataFrame(d, columns = ["sepal_length", "sepal_width", "petal_length", "
scaled_df.head()
```

Out[10]:

	sepal_length	sepal_width	petal_length	petal_width	Y
0	0.782195	0.286805	0.521463	0.130366	0.130366
1	0.784175	0.566349	0.246870	0.058087	0.000000
2	0.701740	0.309591	0.567584	0.216714	0.206394
3	0.736397	0.329730	0.549550	0.186847	0.109910
4	0.775771	0.607125	0.168646	0.033729	0.000000

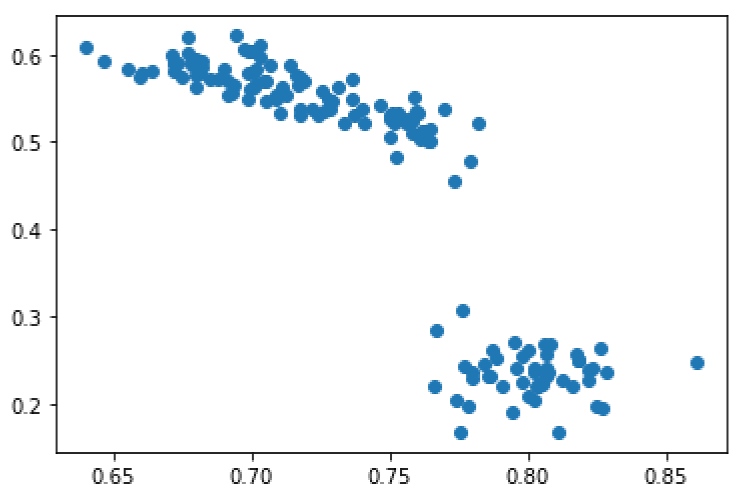
In [11]:

```
print("scatter plot for X1 X3")
plt.scatter(x = scaled_df.sepal_length, y = scaled_df.petal_length)
```

```
scatter plot for X1 X3
```

Out[11]:

```
<matplotlib.collections.PathCollection at 0x206a9999b20>
```

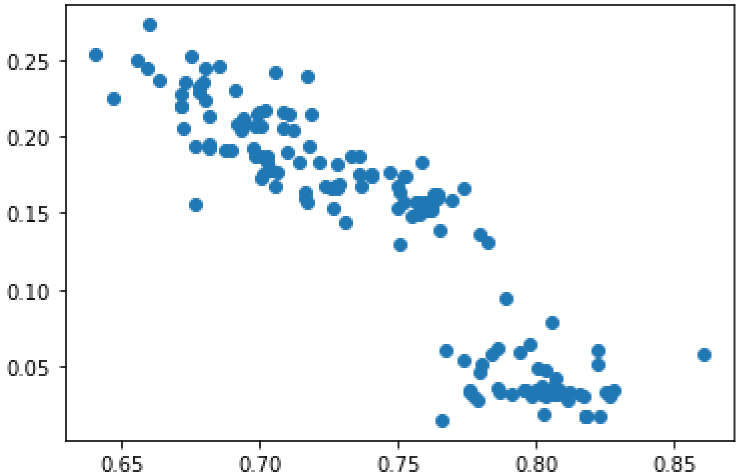


In [12]:

```
print("scatter plot for X1 X4")
plt.scatter(x = scaled_df.sepal_length, y = scaled_df.petal_width)
```

```
scatter plot for X1 X4
```

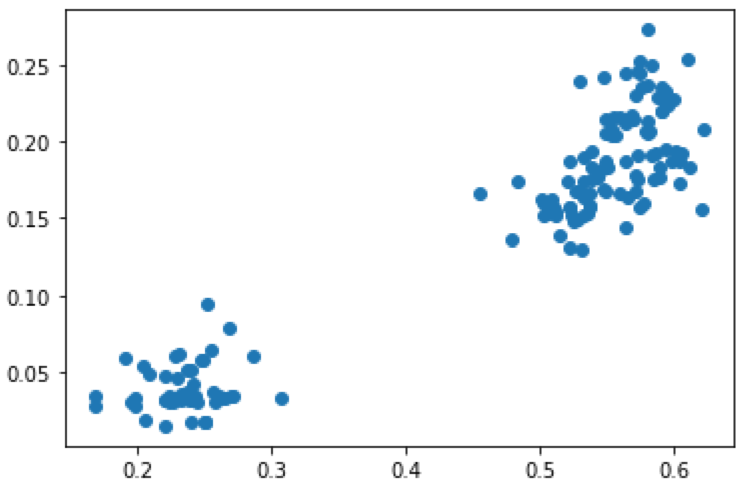
Out[12]: <matplotlib.collections.PathCollection at 0x206a9a0d700>



In [13]:
```
print("scatter plot for X3 X4")
plt.scatter(x = scaled_df.petal_length, y = scaled_df.petal_width)
```

```
scatter plot for X3 X4
```

Out[13]: <matplotlib.collections.PathCollection at 0x206a9a74160>



The three different classes are linearly separable.

Q5-d

decison tree on X1, X3

In [14]:
```
X = df.loc[:,['sepal_length','petal_length']]
Y = df.loc[:,'Y']
```

In [15]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=1
```

In [16]:
```
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,Y_train)
Y_pred = clf.predict(X_test)
```

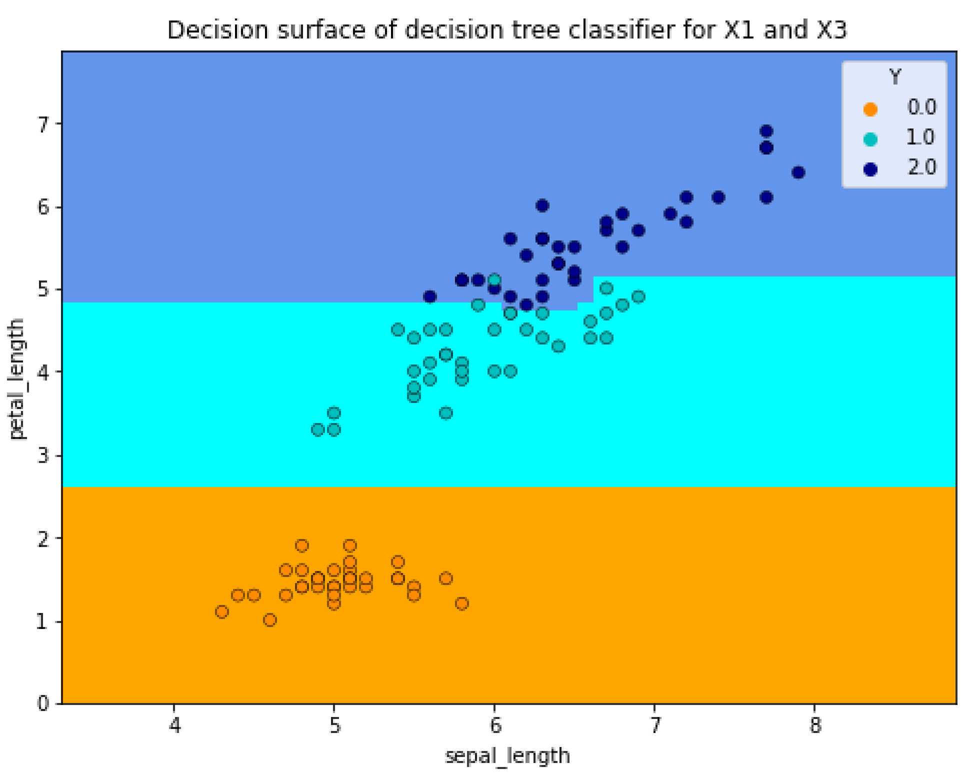
In [17]:

```
print(f'Accuracy for decision tree classifier between X1 and X3 is: {metrics.accuracy_s
```

```
Accuracy for decision tree classifier between X1 and X3 is: 0.911
```

In [18]:

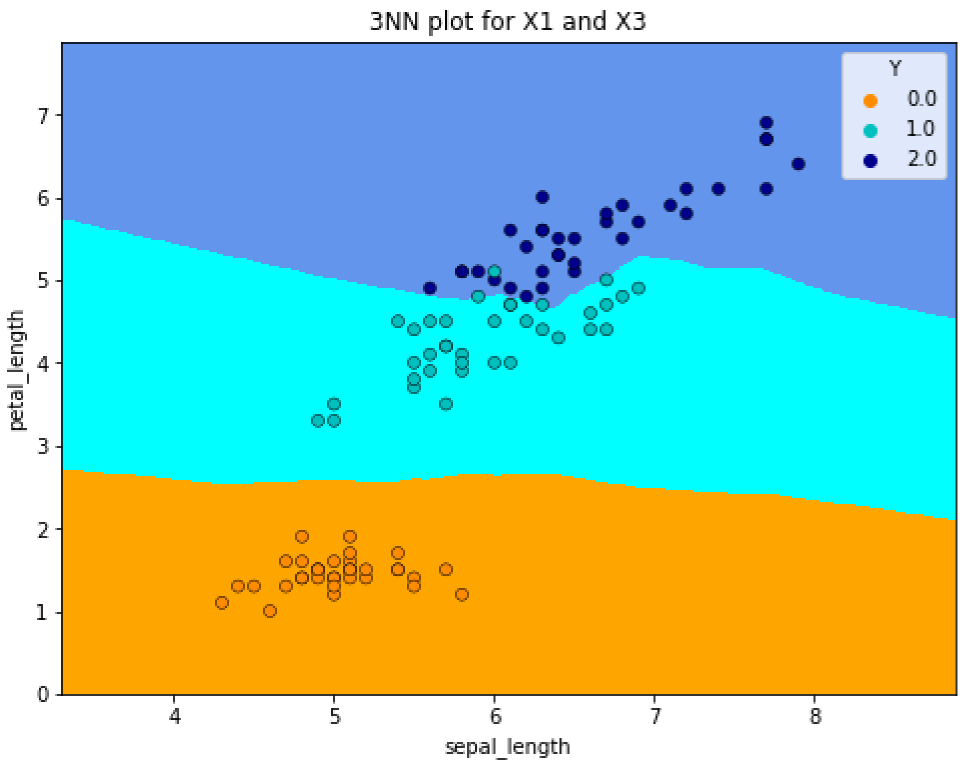
```
h = 0.02
cmap_light = ListedColormap(["orange", "cyan", "cornflowerblue"])
cmap_bold = ["darkorange", "c", "darkblue"]
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)
sns.scatterplot(
        x=X_train.iloc[:, 0],
        y=X_train.iloc[:, 1],
        hue=Y_train,
        palette=cmap_bold,
        alpha=1.0,
        edgecolor="black",
    )
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Decision surface of decision tree classifier for X1 and X3")
plt.xlabel("sepal_length")
plt.ylabel('petal_length')

plt.show()
```



KNN for X1, X3

In [19]:

```
# we create an instance of Neighbours Classifier and fit the data.
clf = KNeighborsClassifier(n_neighbors = 3)
clf.fit(X_train, Y_train)

# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, x_max]x[y_min, y_max].
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)

# Plot also the training points
sns.scatterplot(
    x=X_train.iloc[:, 0],
    y=X_train.iloc[:, 1],
    hue=Y_train,
    palette=cmap_bold,
    alpha=1.0,
    edgecolor="black",
)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("3NN plot for X1 and X3")
plt.xlabel("sepal_length")
plt.ylabel('petal_length')

plt.show()
Y_pred = clf.predict(X_test)
print(f'Accuracy for KNN classifier for X1 and X3 is: {metrics.accuracy_score(Y_test, Y
```




```
Accuracy for KNN classifier for X1 and X3 is: 0.911
```

SVM for X1, X3

In [20]:

```
clf = svm.SVC(gamma = 2, C = 1)
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)
```

In [21]:

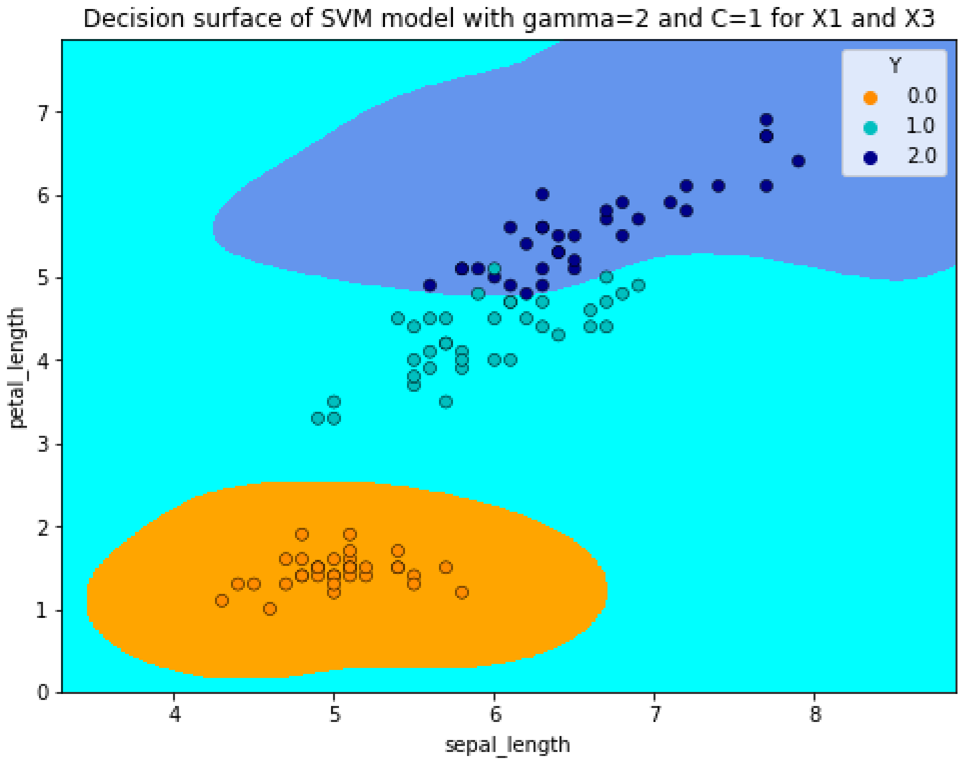
```
print(f'Accuracy for SVM classifier for X1 and X3 is: {metrics.accuracy_score(Y_test, Y
```

```
Accuracy for SVM classifier for X1 and X3 is: 0.933
```

In [22]:

```
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)
sns.scatterplot(
        x=X_train.iloc[:, 0],
        y=X_train.iloc[:, 1],
        hue=Y_train,
        palette=cmap_bold,
        alpha=1.0,
        edgecolor="black",
    )
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Decision surface of SVM model with gamma=2 and C=1 for X1 and X3")
plt.xlabel("sepal_length")
```

```
plt.ylabel('petal_length')

plt.show()
```



Naive Bayers Classifier for X1, X3

In [23]:

```
clf = GaussianNB()
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)
```

In [24]:

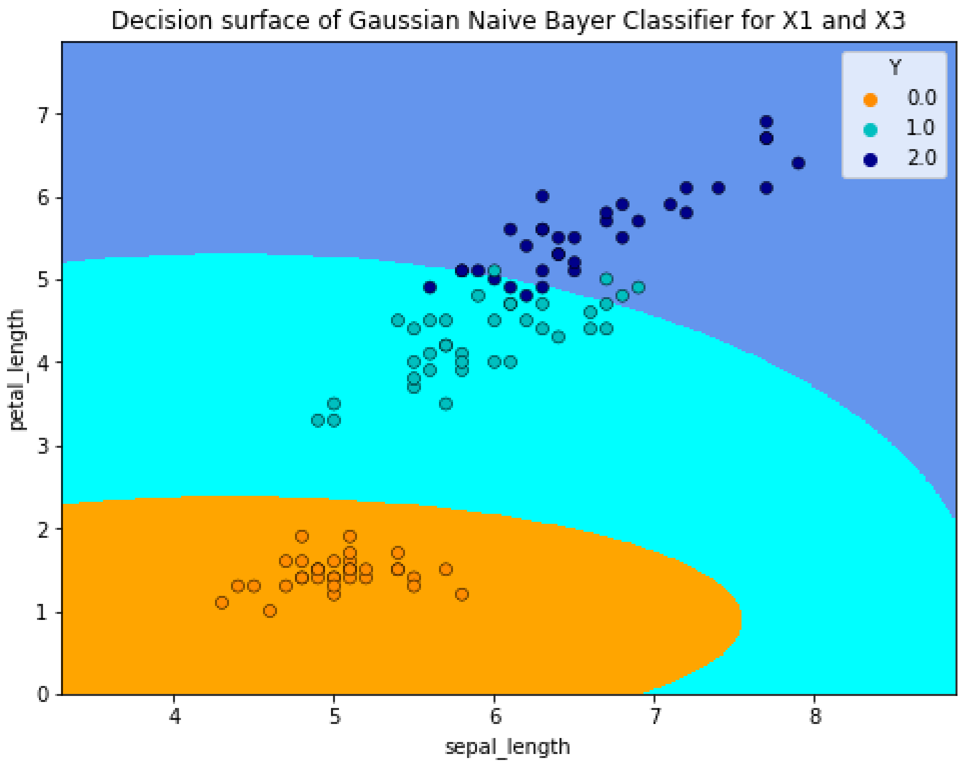
```
print(f'Accuracy for Naive Bayers classifier for X1 and X3 is: {metrics.accuracy_score(
```

```
Accuracy for Naive Bayers classifier for X1 and X3 is: 0.889
```

In [25]:

```
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)
sns.scatterplot(
        x=X_train.iloc[:, 0],
        y=X_train.iloc[:, 1],
        hue=Y_train,
        palette=cmap_bold,
        alpha=1.0,
        edgecolor="black",
    )
plt.xlim(xx.min(), xx.max())
```

```
plt.ylim(yy.min(), yy.max())
plt.title("Decision surface of Gaussian Naive Bayer Classifier for X1 and X3")
plt.xlabel("sepal_length")
plt.ylabel('petal_length')

plt.show()
```



Adaboost Classifier with Decision Tree for X1, X4

In [26]:

```
clf = DecisionTreeClassifier(max_depth = 3)
abc =AdaBoostClassifier(n_estimators=30, base_estimator=clf)
```

In [27]:

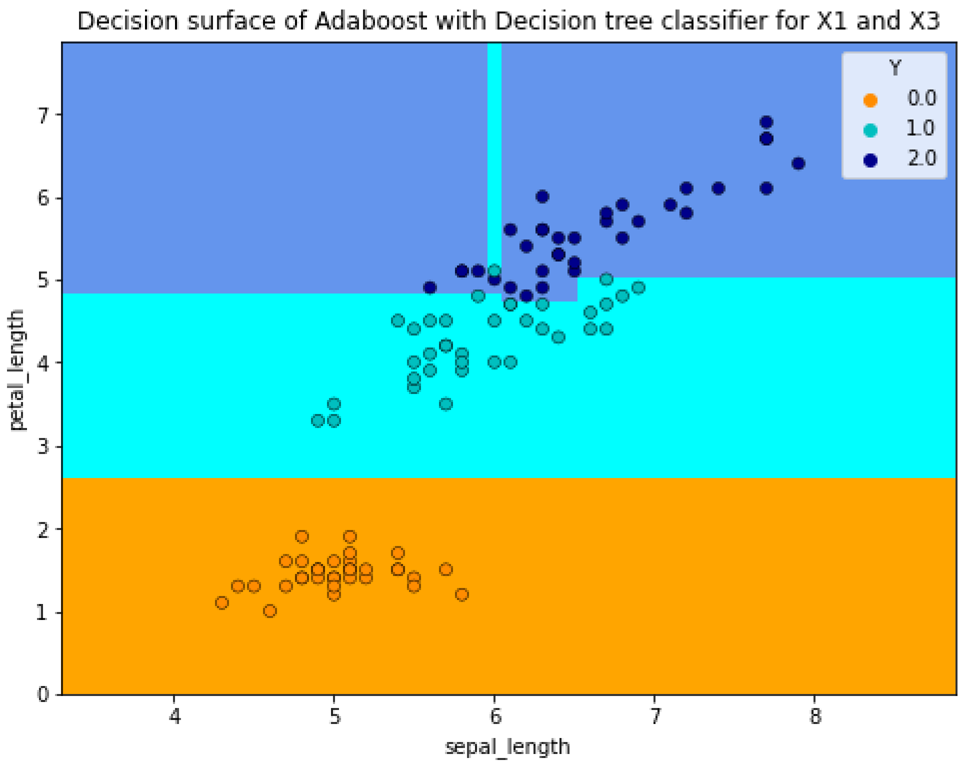
```
abc.fit(X_train, Y_train)
Y_pred = abc.predict(X_test)
print(f'Accuracy for Adaboost Classifier with Decision Tree for X1 and X3 is: {metrics.
```

```
Accuracy for Adaboost Classifier with Decision Tree for X1 and X3 is: 0.933
```

In [28]:

```
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = abc.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)
sns.scatterplot(
        x=X_train.iloc[:, 0],
        y=X_train.iloc[:, 1],
        hue=Y_train,
        palette=cmap_bold,
```

```
        alpha=1.0,
        edgecolor="black",
    )
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Decision surface of Adaboost with Decision tree classifier for X1 and X3")
plt.xlabel("sepal_length")
plt.ylabel('petal_length')

plt.show()
```



decison tree on X1, X4

In [29]:

```
X = df.loc[:,['sepal_length','petal_width']]
Y = df.loc[:,'Y']
```

In [30]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=1
```

In [31]:

```
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,Y_train)
Y_pred = clf.predict(X_test)
```

In [32]:

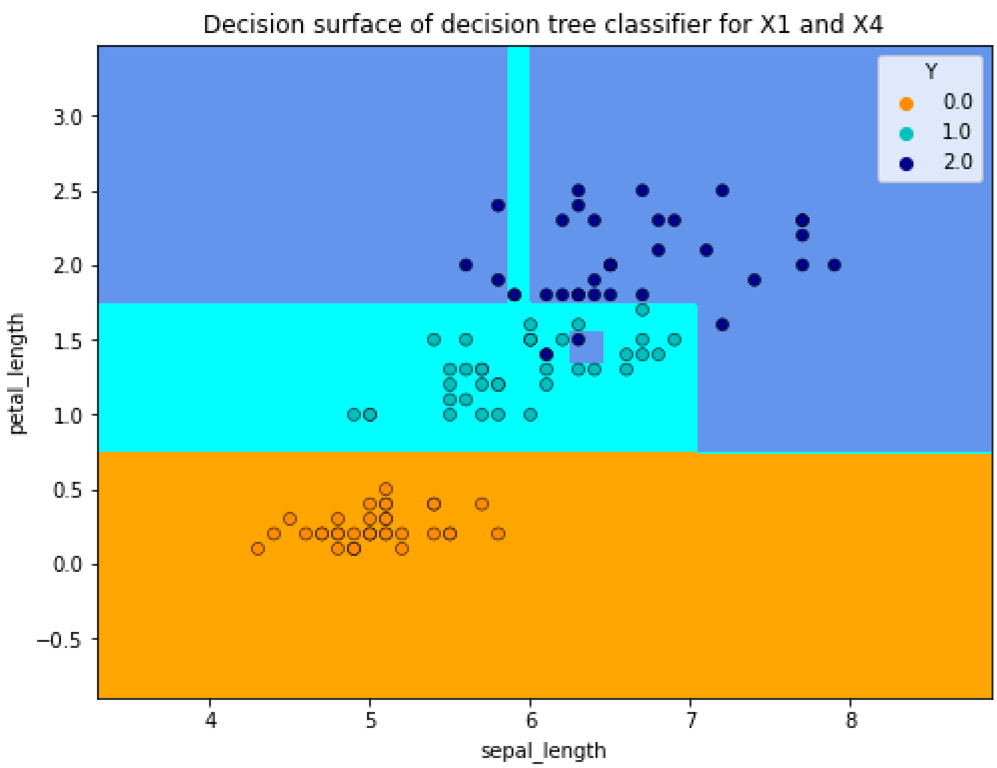
```
print(f'Accuracy for decision tree classifier between X1 and X4 is: {metrics.accuracy_s
```

```
Accuracy for decision tree classifier between X1 and X4 is: 0.911
```

In [33]:

```
h = 0.02
cmap_light = ListedColormap(["orange", "cyan", "cornflowerblue"])
cmap_bold = ["darkorange", "c", "darkblue"]
```

```
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)
sns.scatterplot(
        x=X_train.iloc[:, 0],
        y=X_train.iloc[:, 1],
        hue=Y_train,
        palette=cmap_bold,
        alpha=1.0,
        edgecolor="black",
    )
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Decision surface of decision tree classifier for X1 and X4")
plt.xlabel("sepal_length")
plt.ylabel('petal_length')

plt.show()
```

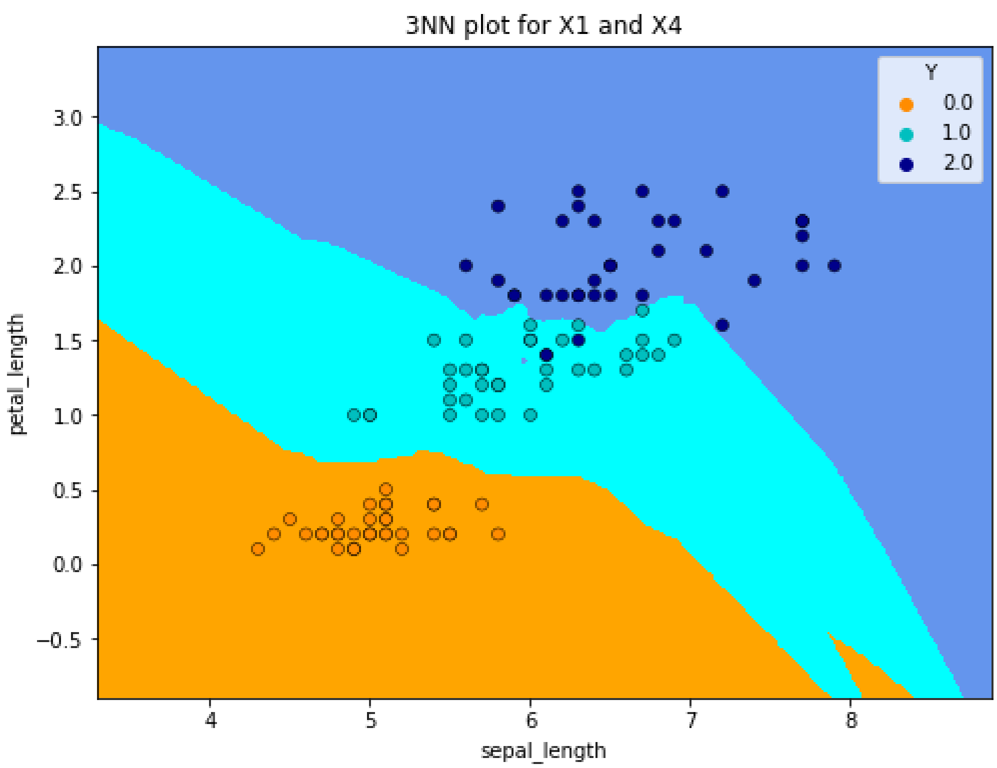


KNN for X1, X4

In [34]:

```
# we create an instance of Neighbours Classifier and fit the data.
clf = KNeighborsClassifier(n_neighbors = 3)
clf.fit(X_train, Y_train)

# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, x_max]x[y_min, y_max].
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
```

```
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)

# Plot also the training points
sns.scatterplot(
    x=X_train.iloc[:, 0],
    y=X_train.iloc[:, 1],
    hue=Y_train,
    palette=cmap_bold,
    alpha=1.0,
    edgecolor="black",
)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("3NN plot for X1 and X4")
plt.xlabel("sepal_length")
plt.ylabel('petal_length')

plt.show()
Y_pred = clf.predict(X_test)
print(f'Accuracy for KNN classifier for X1 and X4 is: {metrics.accuracy_score(Y_test, Y
```




```
Accuracy for KNN classifier for X1 and X4 is: 0.978
```

SVM for X1, X4

In [35]:

```
clf = svm.SVC(gamma = 2, C = 1)
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)
```

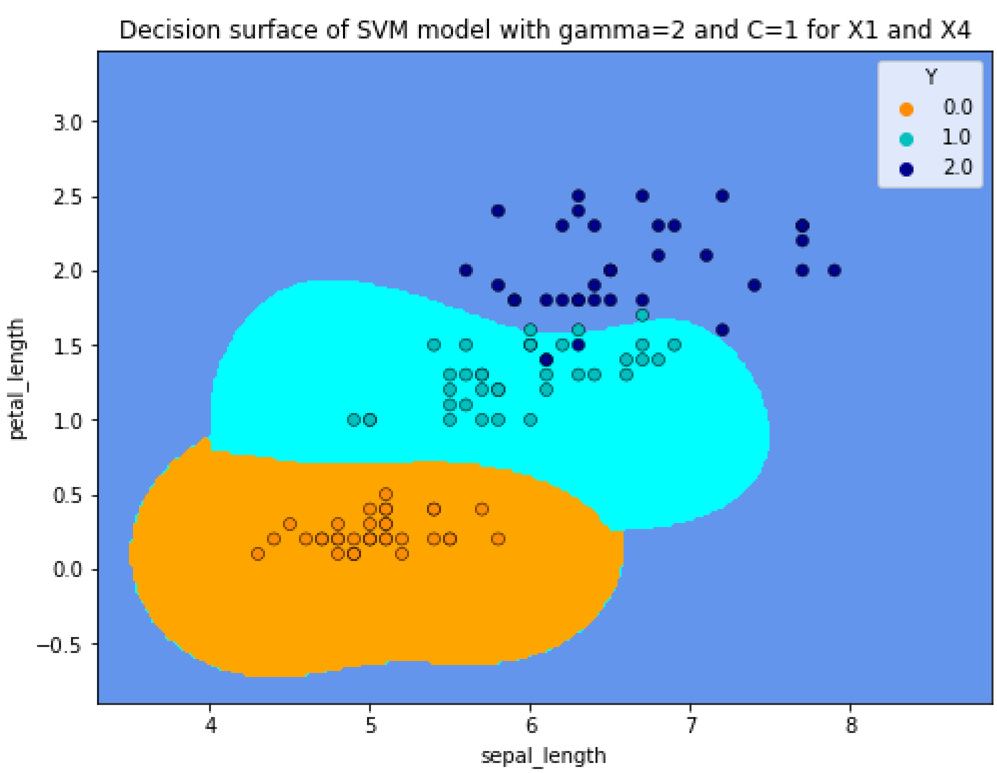
In [36]:

```
print(f'Accuracy for SVM classifier for X1 and X4 is: {metrics.accuracy_score(Y_test, Y
```

```
Accuracy for SVM classifier for X1 and X4 is: 0.978
```

In [37]:

```
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)
sns.scatterplot(
        x=X_train.iloc[:, 0],
        y=X_train.iloc[:, 1],
        hue=Y_train,
        palette=cmap_bold,
        alpha=1.0,
        edgecolor="black",
    )
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Decision surface of SVM model with gamma=2 and C=1 for X1 and X4")
plt.xlabel("sepal_length")
plt.ylabel('petal_length')

plt.show()
```



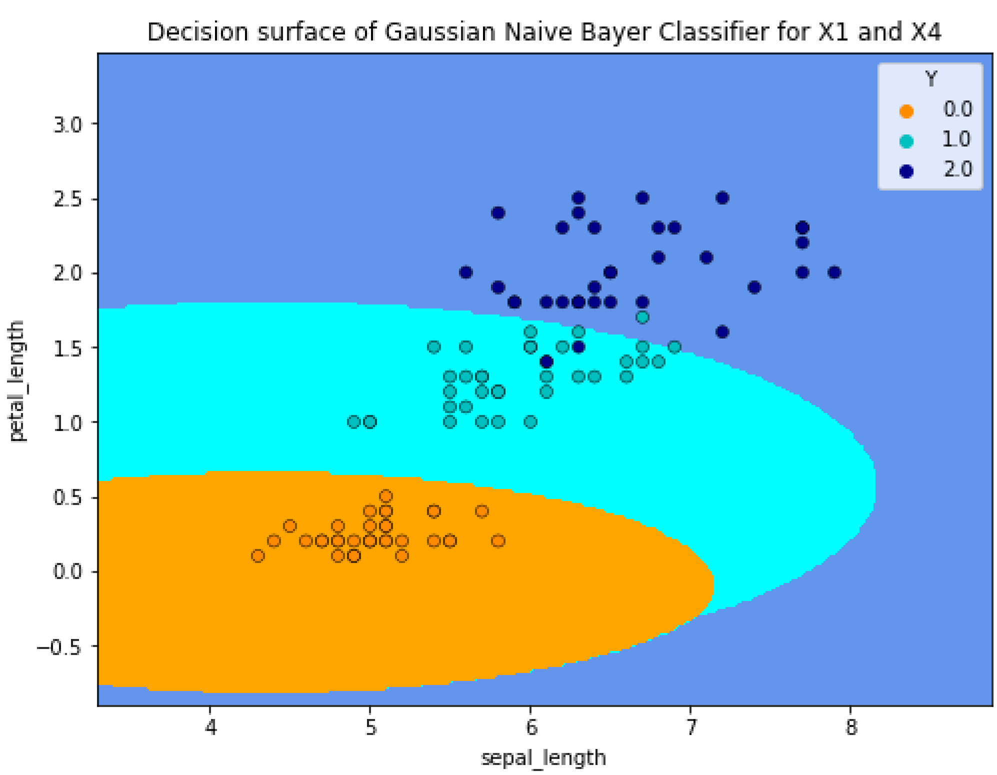
Naive Bayers Classifier for X1, X4

In [38]:
```python
clf = GaussianNB()
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)
```

In [39]:
```python
print(f'Accuracy for Naive Bayers classifier for X1 and X4 is: {metrics.accuracy_score(
```

```
Accuracy for Naive Bayers classifier for X1 and X4 is: 0.978
```

In [40]:
```python
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)
sns.scatterplot(
        x=X_train.iloc[:, 0],
        y=X_train.iloc[:, 1],
        hue=Y_train,
        palette=cmap_bold,
        alpha=1.0,
        edgecolor="black",
    )
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Decision surface of Gaussian Naive Bayer Classifier for X1 and X4")
plt.xlabel("sepal_length")
plt.ylabel('petal_length')

plt.show()
```



Adaboost Classifier with Decision Tree for X1, X4

In [41]:

```
clf = DecisionTreeClassifier(max_depth = 3)
abc =AdaBoostClassifier(n_estimators=30, base_estimator=clf)
```

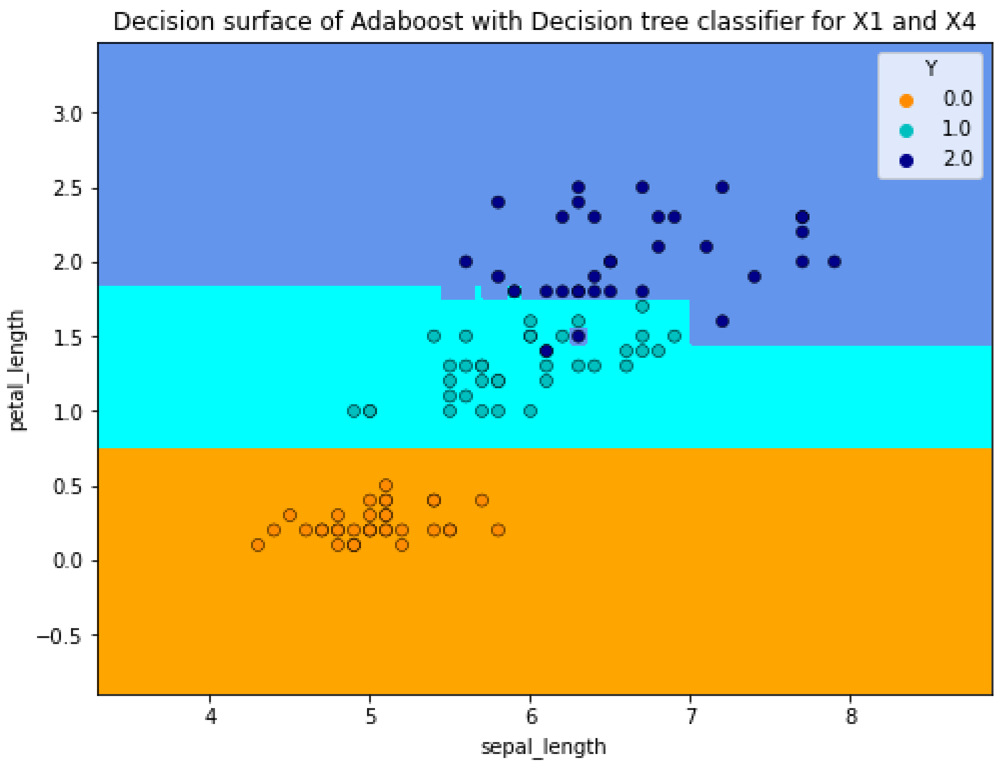
In [42]:

```
abc.fit(X_train, Y_train)
Y_pred = abc.predict(X_test)
print(f'Accuracy for Adaboost Classifier with Decision Tree for X1 and X4 is: {metrics.
```

```
Accuracy for Adaboost Classifier with Decision Tree for X1 and X4 is: 0.956
```

In [43]:

```
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = abc.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)
sns.scatterplot(
        x=X_train.iloc[:, 0],
        y=X_train.iloc[:, 1],
        hue=Y_train,
        palette=cmap_bold,
        alpha=1.0,
        edgecolor="black",
    )
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Decision surface of Adaboost with Decision tree classifier for X1 and X4")
plt.xlabel("sepal_length")
plt.ylabel('petal_length')

plt.show()
```



decison tree on X3, X4

In [44]:
```
X = df.loc[:,['petal_length','petal_width']]
Y = df.loc[:,'Y']
```

In [45]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=1
```

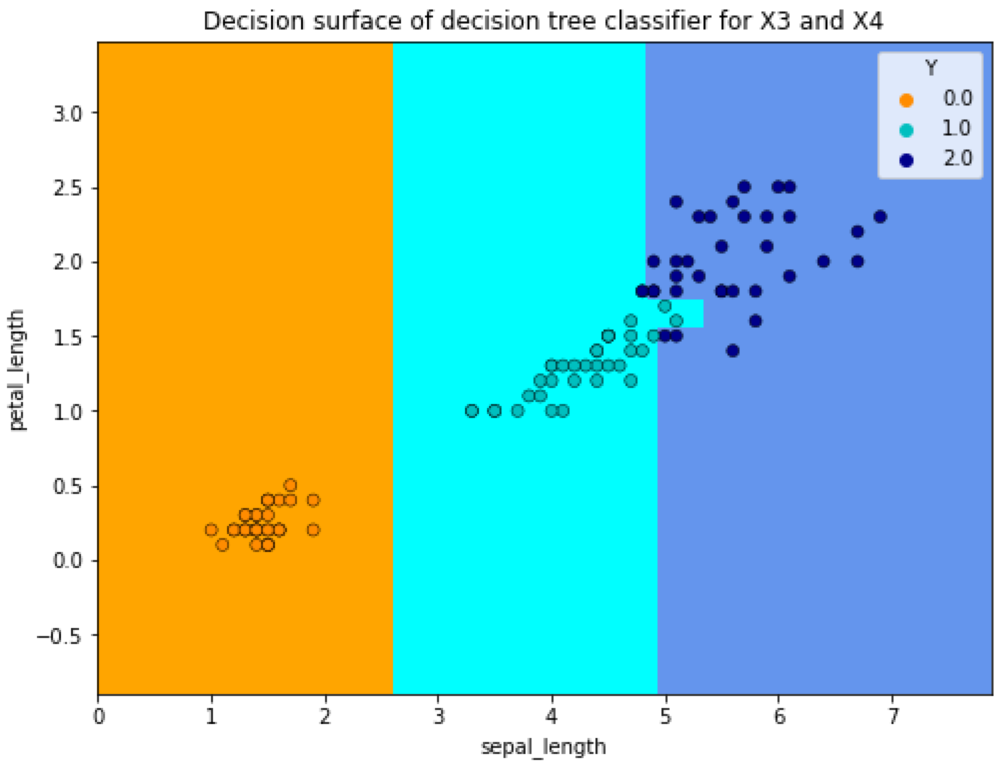
In [46]:
```
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,Y_train)
Y_pred = clf.predict(X_test)
```

In [47]:
```
print(f'Accuracy for decision tree classifier between X3 and X4 is: {metrics.accuracy_s
```

```
Accuracy for decision tree classifier between X3 and X4 is: 0.956
```

In [48]:
```
h = 0.02
cmap_light = ListedColormap(["orange", "cyan", "cornflowerblue"])
cmap_bold = ["darkorange", "c", "darkblue"]
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)
sns.scatterplot(
        x=X_train.iloc[:, 0],
```

```
        y=X_train.iloc[:, 1],
        hue=Y_train,
        palette=cmap_bold,
        alpha=1.0,
        edgecolor="black",
    )
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Decision surface of decision tree classifier for X3 and X4")
plt.xlabel("sepal_length")
plt.ylabel('petal_length')

plt.show()
```

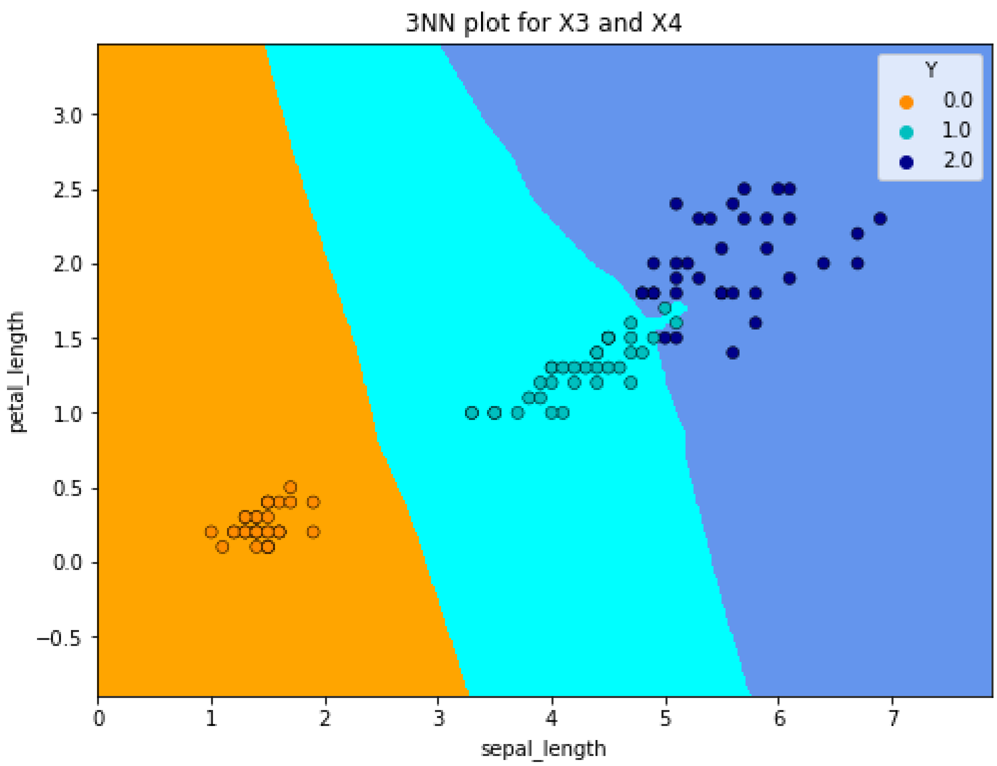


KNN for X3, X4

In [49]:

```
# we create an instance of Neighbours Classifier and fit the data.
clf = KNeighborsClassifier(n_neighbors = 3)
clf.fit(X_train, Y_train)

# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, x_max]x[y_min, y_max].
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)

# Plot also the training points
sns.scatterplot(
```

```
    x=X_train.iloc[:, 0],
    y=X_train.iloc[:, 1],
    hue=Y_train,
    palette=cmap_bold,
    alpha=1.0,
    edgecolor="black",
)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("3NN plot for X3 and X4")
plt.xlabel("sepal_length")
plt.ylabel('petal_length')

plt.show()
Y_pred = clf.predict(X_test)
print(f'Accuracy for KNN classifier for X3 and X4 is: {metrics.accuracy_score(Y_test, Y
```




```
Accuracy for KNN classifier for X3 and X4 is: 0.978
```

SVM for X3, X4

In [50]:

```
clf = svm.SVC(gamma = 2, C = 1)
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)
```

In [51]:

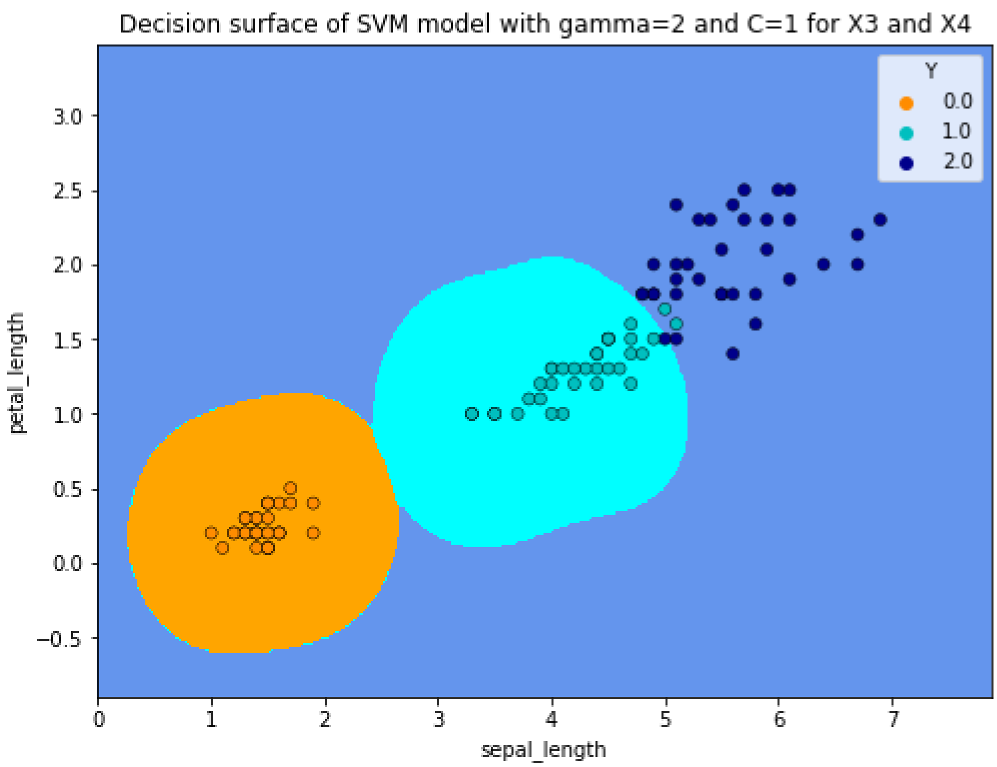
```
print(f'Accuracy for SVM classifier for X3 and X4 is: {metrics.accuracy_score(Y_test, Y
```

```
Accuracy for SVM classifier for X3 and X4 is: 0.978
```

In [52]:

```
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
```

```
# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)
sns.scatterplot(
        x=X_train.iloc[:, 0],
        y=X_train.iloc[:, 1],
        hue=Y_train,
        palette=cmap_bold,
        alpha=1.0,
        edgecolor="black",
    )
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Decision surface of SVM model with gamma=2 and C=1 for X3 and X4")
plt.xlabel("sepal_length")
plt.ylabel('petal_length')

plt.show()
```



Naive Bayers Classifier for X3, X4

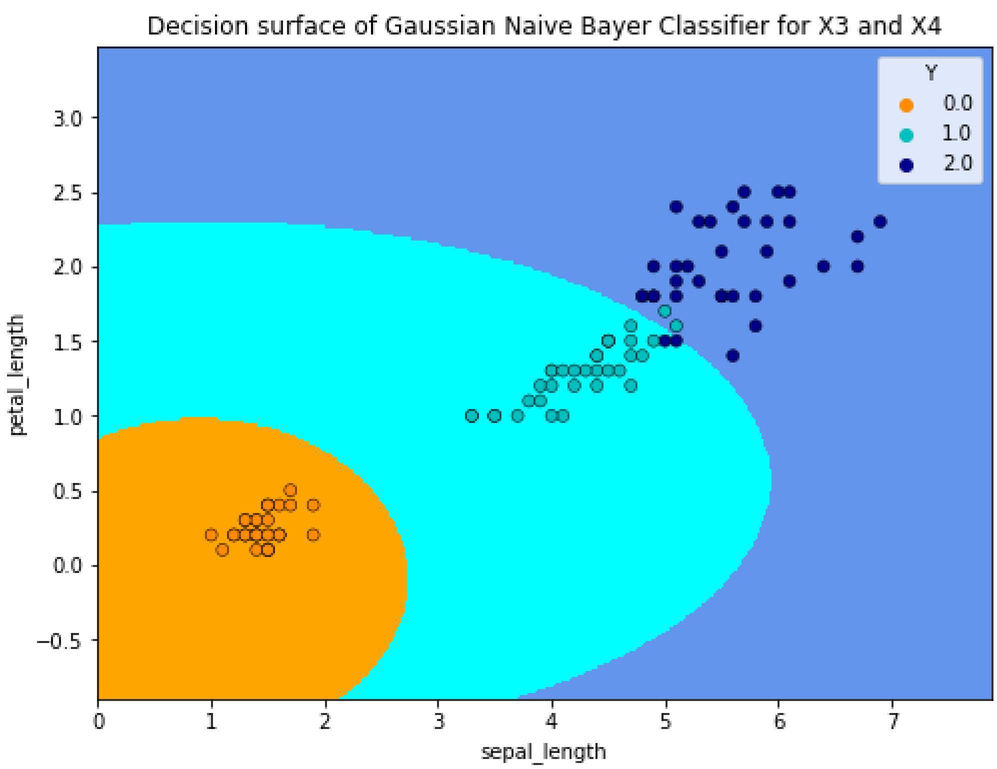
In [53]:
```
clf = GaussianNB()
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)
```

In [54]:
```
print(f'Accuracy for Naive Bayers classifier for X3 and X4 is: {metrics.accuracy_score(
```

```
Accuracy for Naive Bayers classifier for X3 and X4 is: 0.978
```

In [55]:
```
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
```

```
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)
sns.scatterplot(
        x=X_train.iloc[:, 0],
        y=X_train.iloc[:, 1],
        hue=Y_train,
        palette=cmap_bold,
        alpha=1.0,
        edgecolor="black",
    )
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Decision surface of Gaussian Naive Bayer Classifier for X3 and X4")
plt.xlabel("sepal_length")
plt.ylabel('petal_length')

plt.show()
```



Adaboost Classifier with Decision Tree for X3, X4

In [56]:

```
clf = DecisionTreeClassifier(max_depth = 3)
abc =AdaBoostClassifier(n_estimators=30, base_estimator=clf)
```

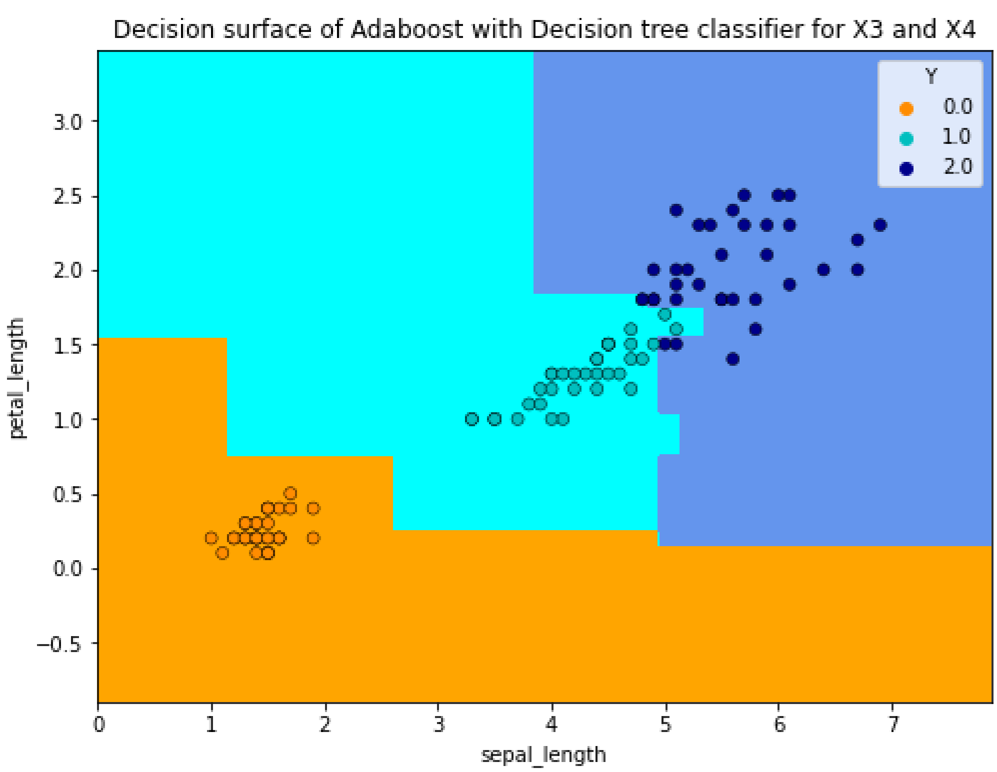
In [57]:

```
abc.fit(X_train, Y_train)
Y_pred = abc.predict(X_test)
print(f'Accuracy for Adaboost Classifier with Decision Tree for X3 and X4 is: {metrics.
```

```
Accuracy for Adaboost Classifier with Decision Tree for X3 and X4 is: 0.956
```

In [58]:

```
x_min, x_max = X_train.iloc[:, 0].min() - 1, X_train.iloc[:, 0].max() + 1
y_min, y_max = X_train.iloc[:, 1].min() - 1, X_train.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = abc.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=cmap_light)
sns.scatterplot(
        x=X_train.iloc[:, 0],
        y=X_train.iloc[:, 1],
        hue=Y_train,
        palette=cmap_bold,
        alpha=1.0,
        edgecolor="black",
    )
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Decision surface of Adaboost with Decision tree classifier for X3 and X4")
plt.xlabel("sepal_length")
plt.ylabel('petal_length')

plt.show()
```



Q5-d

Decision Tree Classifier's advantage is its interpretability and that there is no need for feature scaling. Also, decision tree classifier works on both linear and nonlinear problems.

Decision Tree Classifier's disadvantage is its poor results on very small datasets. Also, overfitting can easily occur.

K Nearest Neighbours Classifier's advantage is that it's simple to understand, fast and efficient.

K Nearest Neighbours Classifier's advantage is that we need to manually choose the number of neighbours ‘k’.

SVM's advantage is its high performance on nonlinear problems. And it's not biased by outliers. It is alsp not sensitive to overfitting.

SVM's disadvantage is that it is not the best choice for large number of features. And it's often more complex.

Naive Bayers Classifier's advantage is that it's efficient. And it's not biased by outliers. It works on nonlinear problems and it's a probabilistic approach.

Naive Bayers Classifier's disadvantage is that it's based in the assumption that the features have same statistical relevance.

Adaboost classifier's advantage is that it is easier to use with less need for tweaking parameters unlike algorithms like SVM.

Adaboost Classifier's disadvantage is that boosting technique learns progressively, it is important to ensure that you have quality data. AdaBoost is also extremely sensitive to Noisy data and outliers so if you do plan to use AdaBoost then it is highly recommended to eliminate them.