FIND S

In [2]:

```
import pandas as pd

data = pd.read_csv('enjoysport.csv')
print(data)

attributes = ['sky', 'temp', 'humidity', 'wind', 'water', 'forcast']
target = ['yes','yes','no','yes']

len_attribute = len(attributes)

h = ['0']*len_attribute
for i in range(len(target)):
    if target[i]=='yes':
        for j in range(len_attribute):
            if h[j]=='0':
                h[j]=data.iloc[i][j]
            if h[j]!=data.iloc[i][j]:
                h[j]='?'
    print(i+1, h)
```

```
       a     b       c       d     e       f    g
0  sunny  warm  normal  strong  warm    same  Yes
1  sunny  warm    high  strong  warm    same  Yes
2  rainy  cold    high  strong  warm  change   No
3  sunny  warm    high  strong  cool  change  Yes
1 ['sunny', 'warm', 'normal', 'strong', 'warm', 'same']
2 ['sunny', 'warm', '?', 'strong', 'warm', 'same']
3 ['sunny', 'warm', '?', 'strong', 'warm', 'same']
4 ['sunny', 'warm', '?', 'strong', '?', '?']
```

Candidate Elimintaion

In [5]:

```
import pandas as pd
import numpy as np

data = pd.read_csv('enjoysport.csv')

concepts = np.array(data.iloc[:,:-1])
target = np.array(data.iloc[:,-1])

def solve(target,concepts):
    spec_h = concepts[0].copy()

    gen_h = [['?' for i in range(len(spec_h))]for j in range(len(spec_h))]

    for i,h in enumerate(concepts):
        if target[i]=='Yes':
            for j in range(len(spec_h)):
                if h[j] != spec_h[j] :
                    spec_h[j] = '?'
                    gen_h[j][j] = '?'
        if target[i]=='No':
            for j in range(len(spec_h)):
                if h[j] != spec_h[j]:
                    gen_h[j][j] = spec_h[j]
                else:
                    gen_h[j][j] = '?'

        print(i+1)
        print("Specific :" , spec_h)
        print("General :" , gen_h)

    while ['?','?','?','?','?','?'] in gen_h:
        gen_h.remove(['?','?','?','?','?','?'])

    return spec_h,gen_h

final_spec,final_gen = solve(target,concepts)

print("Final Specific: ",final_spec)
print("Final General :",final_gen)
```

```
1
Specific : ['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
General : [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?']]
2
Specific : ['sunny' 'warm' '?' 'strong' 'warm' 'same']
General : [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?']]
3
Specific : ['sunny' 'warm' '?' 'strong' 'warm' 'same']
General : [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
```

```
'same']]
4
Specific : ['sunny' 'warm' '?' 'strong' '?' '?']
General : [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?']]
Final Specific:  ['sunny' 'warm' '?' 'strong' '?' '?']
Final General : [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm',
'?', '?', '?', '?']]
```

BPP

In [8]:

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,accuracy_score

iris = load_iris()
value = iris.data
target = iris.target

ss = StandardScaler()
value = ss.fit_transform(value)

x_train,x_test,y_train,y_test = train_test_split(value,target,test_size=0.3)

n=1000
loss_cur = 999

clf = MLPClassifier(hidden_layer_sizes=(4,3),activation='logistic',solver='sgd',lea

for i in range(n):
    clf.fit(x_train,y_train)
    loss_prev = loss_cur
    loss_cur = clf.loss_
    for i in clf.coefs_:
        print(i,end='\n\n')
    if(abs(loss_cur-loss_prev)<0.0001):
        break

y_pred = clf.predict(x_test)

print("Confusion Matrix ", confusion_matrix(y_test,y_pred))
print("Accuracy Score ", accuracy_score(y_test,y_pred))
```

```
 [ 1.5202808   1.55531238 -2.45447772 -1.80191376]]

[[ 2.44194853  2.93415877 -2.22103138]
 [ 2.04266161  4.61106779 -1.6459185 ]
 [-4.61342673 -2.9529716   3.23091693]
 [-3.00494084 -0.8364789   2.54204667]]

[[-2.1577147  -3.40340743  5.94950652]
 [-8.27000059  3.57083759  4.99600291]
 [ 3.37445303  1.90125191 -4.90895619]]

Iteration 191, loss = 0.03575882
[[ 0.2495818   0.43960753  0.37082758  0.24238124]
 [-0.864918   -1.76915411  0.24706248  0.15716671]
 [ 1.20647193  0.97072129 -2.06740746 -1.59727302]
 [ 1.52343401  1.55544235 -2.45924009 -1.80612364]]

[[ 2.44553738  2.93648704 -2.22359382]
 [ 2.04460567  4.61587553 -1.64730391]
 [-4.61947933 -2.95360568  3.23560864]
```

Naive Bayes

In [13]:

```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

le = LabelEncoder()
data = pd.read_csv('tennis.csv')

data_df = pd.DataFrame(data)
data_df_encoded = data_df.apply(le.fit_transform)

values = data_df_encoded.drop(['e'],axis=1)
target = data_df_encoded['e']

x_train,x_test,y_train,y_test = train_test_split(values,target,test_size=0.3)

model = GaussianNB()
model.fit(x_train,y_train)
y_pred = model.predict(x_test)

print(list(y_pred))
print(list(y_test))

print("Confusion Matrix ", confusion_matrix(y_test,y_pred))
print("Accuracy Score ", accuracy_score(y_test,y_pred))
```

```
[2, 2, 2, 2, 2]
[0, 2, 2, 2, 2]
Confusion Matrix  [[0 1]
 [0 4]]
Accuracy Score  0.8
```

Bayesian Network

In [21]:

```
import pandas as pd
from pgmpy.models import BayesianModel
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.inference import VariableElimination

heart = pd.read_csv('data7_heart.csv')
heart = heart.replace('?',np.nan)

model = BayesianModel([('sex', 'trestbps'),
    ('exang', 'trestbps'),
    ('age', 'trestbps'),
    ('age', 'fbs'),
    ('trestbps', 'heartdisease'),
    ('trestbps', 'fbs'),
    ('heartdisease', 'restecg'),
    ('heartdisease', 'thalach'),
    ('heartdisease', 'chol')])

model.fit(heart,estimator=MaximumLikelihoodEstimator)

infer = VariableElimination(model)
q = infer.query(variables=['heartdisease'],evidence={
    'age':67,'sex':1
})

print(q)
```

```
/home/aizwal/.local/lib/python3.8/site-packages/pgmpy/models/BayesianM
odel.py:8: FutureWarning: BayesianModel has been renamed to BayesianNe
twork. Please use BayesianNetwork class, BayesianModel will be removed
in future.
  warnings.warn(
```

Finding Elimination Order: : 100% 2/2 [00:00<00:00, 45.53it/s]

Eliminating: exang: 100% 2/2 [14:38<00:00, 439.43s/it]

```
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.4683 |
+-----------------+---------------------+
| heartdisease(1) |              0.2794 |
+-----------------+---------------------+
| heartdisease(2) |              0.0830 |
+-----------------+---------------------+
| heartdisease(3) |              0.1279 |
+-----------------+---------------------+
| heartdisease(4) |              0.0414 |
+-----------------+---------------------+
```

KNN

In [26]:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

iris = load_iris()
values = iris.data
target = iris.target

x_train,x_test,y_train,y_test = train_test_split(values,target,test_size=0.3)

model = KNeighborsClassifier()
model.fit(x_train,y_train)
y_pred = model.predict(x_test)

diff = y_pred-y_test

print(diff)

count=0
for i in diff:
    if i!=0:
        count+=1

print(count)
```

```
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 0 0 0 0 0 0 0 0]
2
```

KMEANS AND EM

In [39]:

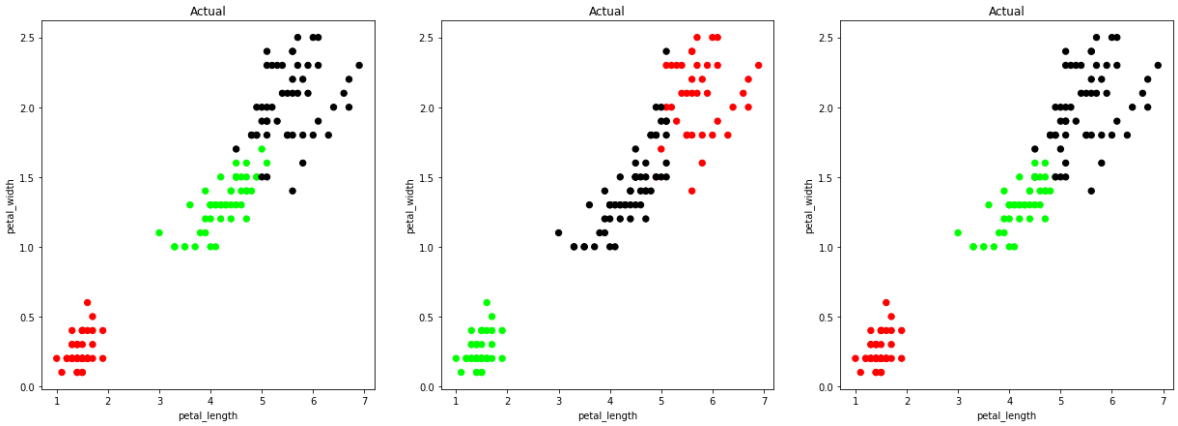
```
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.metrics import confusion_matrix,accuracy_score
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

iris = load_iris()
x = pd.DataFrame(iris.data)
x.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']

y = pd.DataFrame(iris.target)
y.columns = ['Targets']

#KMEANS
model = KMeans(n_clusters=3)
model.fit(x)
y_kmean = model.predict(x)


print("Confusion Matrix ", confusion_matrix(y,y_kmean))
print("Accuracy Score ", accuracy_score(y,y_kmean))

gm = GaussianMixture(n_components=3,random_state=0)
gm.fit(x)
y_gm = gm.predict(x)


print("Confusion Matrix ", confusion_matrix(y,y_gm))
print("Accuracy Score ", accuracy_score(y,y_gm))

#Plot

plt.figure(figsize=(21,7))
colormap = np.array(['Red','Lime','Black'])

plt.subplot(1,3,1)
plt.scatter(x.petal_length,x.petal_width,c=colormap[y.Targets],s=40)
plt.title("Actual")
plt.xlabel("petal_length")
plt.ylabel("petal_width")

plt.subplot(1,3,2)
plt.scatter(x.petal_length,x.petal_width,c=colormap[y_kmean],s=40)
plt.title("Actual")
plt.xlabel("petal_length")
plt.ylabel("petal_width")

plt.subplot(1,3,3)
plt.scatter(x.petal_length,x.petal_width,c=colormap[y_gm],s=40)
plt.title("Actual")
plt.xlabel("petal_length")
plt.ylabel("petal_width")
```

```
Confusion Matrix  [[ 0 50  0]
 [ 2  0 48]
```

```
 [36  0 14]]
Accuracy Score  0.09333333333333334
Confusion Matrix  [[50  0  0]
 [ 0 45  5]
 [ 0  0 50]]
Accuracy Score  0.9666666666666667
```

Out[39]:

```
Text(0, 0.5, 'petal_width')
```



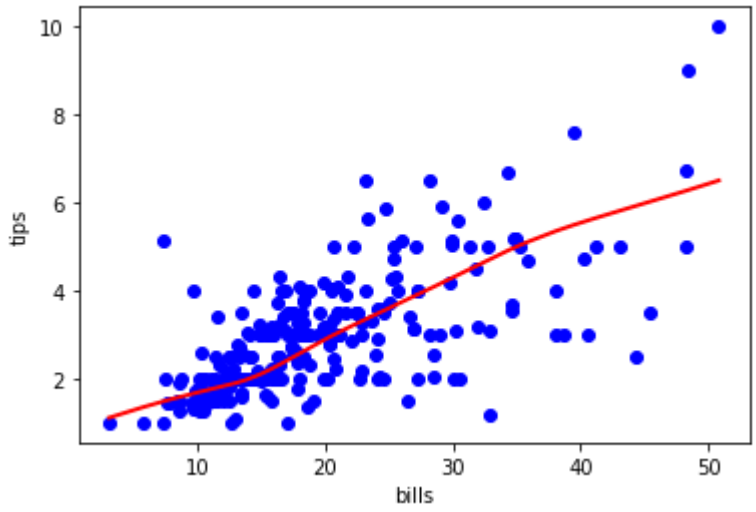
LWR

In [63]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

def kernel(point,xmat,k):
    m,n = np.shape(xmat)
    weights = np.mat(np.eye(m))

    for j in range(m):
        diff = point - xmat[j]
        weights[j,j] = (diff*diff.T)/(-2.0)*(k**2)
    return weights

def localWeight(point,xmat,ymat,k):
    wei = kernel(point,xmat,k)
    W= (xmat.T * wei*xmat).I * (xmat.T*wei*ymat.T)
#     print(W)
    return W

def locallyWeightedRegression(xmat,ymat,k):
    m,n = np.shape(xmat)
    y_pred = np.zeros(m)

    for j in range(m):
        y_pred[j] = xmat[j]*localWeight(xmat[j],xmat,ymat,k)

    return y_pred



data = pd.read_csv('data10_tips.csv')
bill = np.array(data.total_bill)
tip = np.array(data.tip)

mbill = np.mat(bill)
mtip = np.mat(tip)

m = np.shape(mbill)[1]
# print(m)
one = np.mat(np.ones(m))
# print(one)
x = np.hstack((one.T,mbill.T))
# print(x)
y_pred = locallyWeightedRegression(x,mtip,2)

xsorted = x[:,1].argsort(0)
xsort = x[xsorted][:,0]

fig = plt.figure()
ax = fig.add_subplot(1,1,1)

ax.scatter(bill,tip,color='Blue')
ax.plot(xsort[:,1],y_pred[xsorted],color='Red',linewidth=2)
plt.xlabel('bills')
plt.ylabel('tips')
```

Out[63]:

```
Text(0, 0.5, 'tips')
```



In [ ]: