```
import math, random

data = [
[5.1,3.5,1.4,0.2,0],[4.9,3.0,1.4,0.2,0],[5.0,3.4,1.5,0.2,0],
[6.0,2.2,4.0,1.0,1],[5.5,2.3,4.0,1.3,1],[6.5,3.0,5.2,2.0,2],
[6.2,3.4,5.4,2.3,2]
]

random.shuffle(data)
train, test = data[:4], data[4:]

def separate_by_class(dataset):
    d={}
    for r in dataset: d.setdefault(r[-1], []).append(r[:-1])
    return d

def mean_std(dataset):
    return [(sum(col)/len(col), math.sqrt(sum((x-sum(col)/len(col))**2 for x in col)/max(len(col)-1,1)))
            for col in zip(*dataset)]

def summarize_by_class(dataset):
    return {k:mean_std(v) for k,v in separate_by_class(dataset).items()}

def prob(x, m, s): return (1/(math.sqrt(2*math.pi)*s))*math.exp(-(x-m)**2/(2*s**2)) if s>0 else 1

def class_prob(summaries,row):
    probs={}
    for c,s in summaries.items():
        probs[c]=1
        for i,(m,sig) in enumerate(s): probs[c]*=prob(row[i],m,sig)
    return probs
```

```
def predict(summaries,row): return max(class_prob(summaries,row), key=lambda k:
class_prob(summaries,row)[k])


summaries = summarize_by_class(train)

predictions = [predict(summaries,row[:-1]) for row in test]

actual = [row[-1] for row in test]


print("Predicted:", predictions)

print("Actual   :", actual)

acc=sum([predictions[i]==actual[i] for i in range(len(actual))])/len(actual)

print("Accuracy :", acc)
```

OUTPUT: