HW1

problem1

```
In [4]:
import numpy as np
import matplotlib.pyplot as plt

nych = np.genfromtxt('178-hw1-code/data/nyc_housing.txt',delimiter = None)
Y = nych[:,-1]
X = nych[:,0:-1]
```

1.

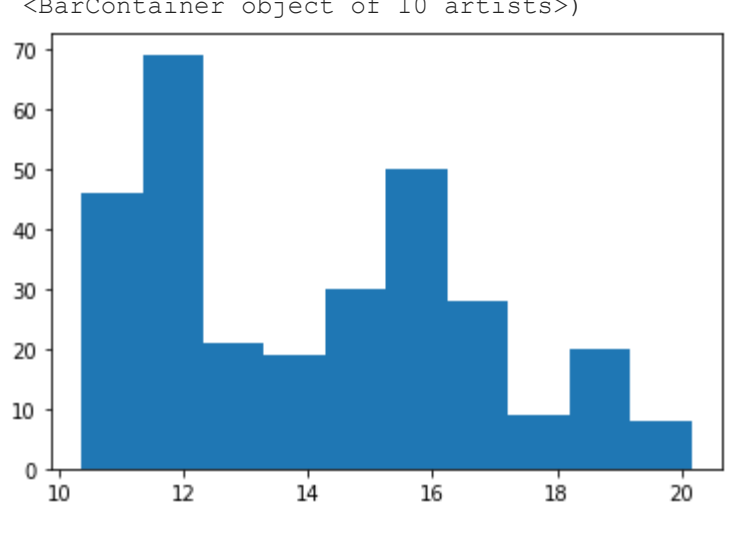
```
In [3]:
X.shape
```

```
Out[3]: (300, 3)
```

2.

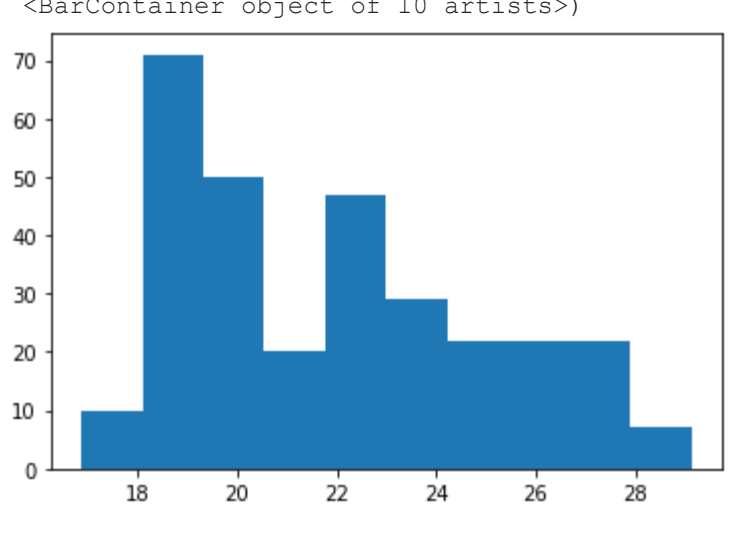
```
In [4]:
plt.hist(X[:,0])
```

```
Out[4]: (array([46., 69., 21., 19., 30., 50., 28.,  9., 20.,  8.]),
 array([10.36632221, 11.34496143, 12.32360065, 13.30223986, 14.28087908,
        15.2595183 , 16.23815751, 17.21679673, 18.19543594, 19.17407516,
        20.15271438]),
 <BarContainer object of 10 artists>)
```



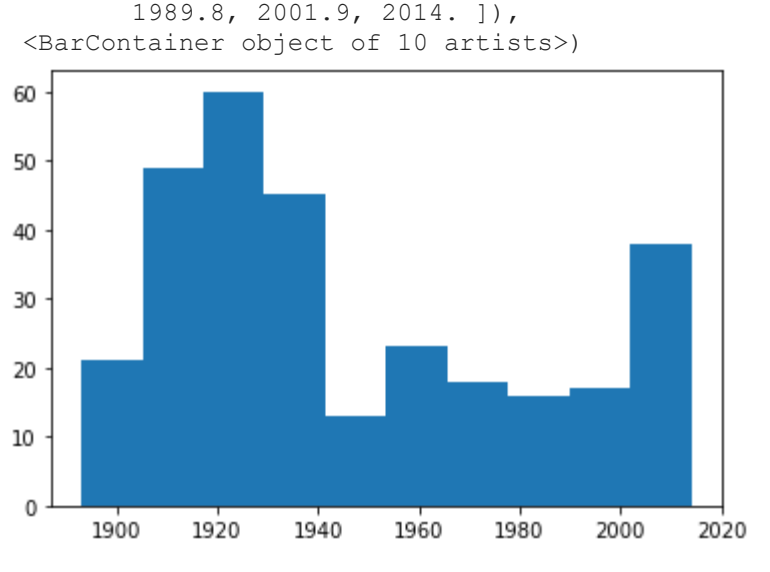
```
In [5]:
plt.hist(X[:,1])
```

```
Out[5]: (array([10., 71., 50., 20., 47., 29., 22., 22., 22.,  7.]),
 array([16.87267488, 18.09779353, 19.32291218, 20.54803083, 21.77314948,
        22.99826813, 24.22338678, 25.44850543, 26.67362408, 27.89874273,
        29.12386138]),
 <BarContainer object of 10 artists>)
```



```
In [6]:
plt.hist(X[:,2])
```

```
Out[6]: (array([21., 49., 60., 45., 13., 23., 18., 16., 17., 38.]),
 array([1893. , 1905.1, 1917.2, 1929.3, 1941.4, 1953.5, 1965.6, 1977.7,
        1989.8, 2001.9, 2014. ]),
 <BarContainer object of 10 artists>)
```



3.

```
In [7]:
my_list = []
for i in range(3):
    my_tuple = (np.mean(X[:,i]),np.std(X[:,i]))
    my_list.append(my_tuple)
my_list
```

```
Out[7]: [(14.118392438424483, 2.569090284260317),
 (21.907116176170856, 2.9785784999947165),
 (1946.3533333333332, 35.39889577687731)]
```

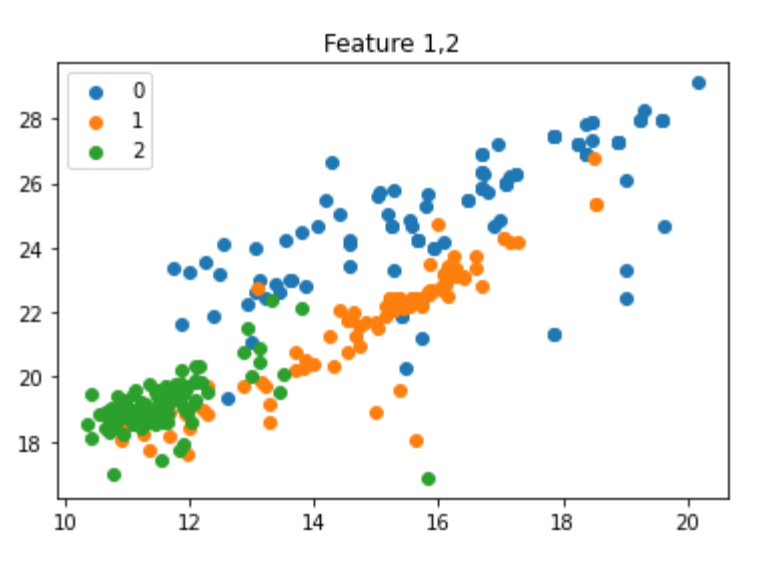
4.

```
In [8]:
nych_0 = nych[nych[:,-1] == 0]
nych_1 = nych[nych[:,-1] == 1]
nych_2 = nych[nych[:,-1] == 2]
```

Plot (1,2)

```
In [9]:
plt.scatter(nych_0[:,0],nych_0[:,1])
plt.scatter(nych_1[:,0],nych_1[:,1])
plt.scatter(nych_2[:,0],nych_2[:,1])
plt.legend([0,1,2])
plt.title('Feature 1,2')
```

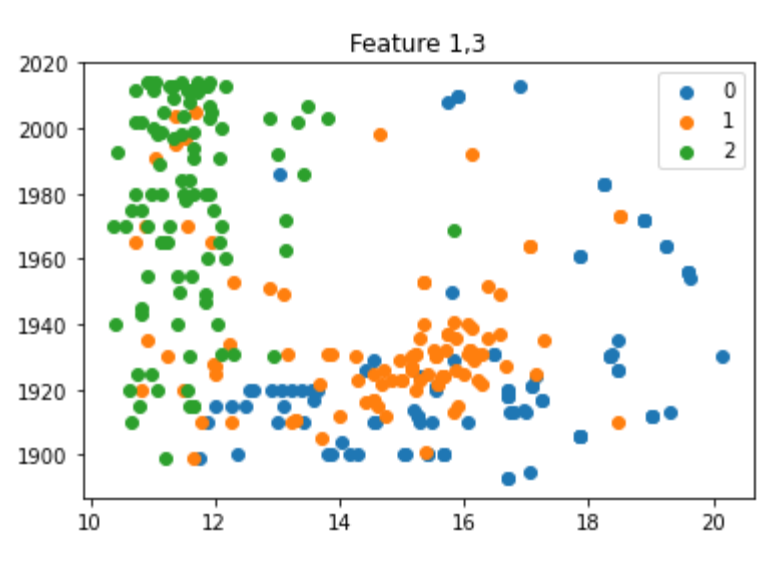
```
Out[9]: Text(0.5, 1.0, 'Feature 1,2')
```



Plot (1,3)

```
In [10]:
plt.scatter(nych_0[:,0],nych_0[:,2])
plt.scatter(nych_1[:,0],nych_1[:,2])
plt.scatter(nych_2[:,0],nych_2[:,2])
plt.legend([0,1,2])
plt.title('Feature 1,3')
```

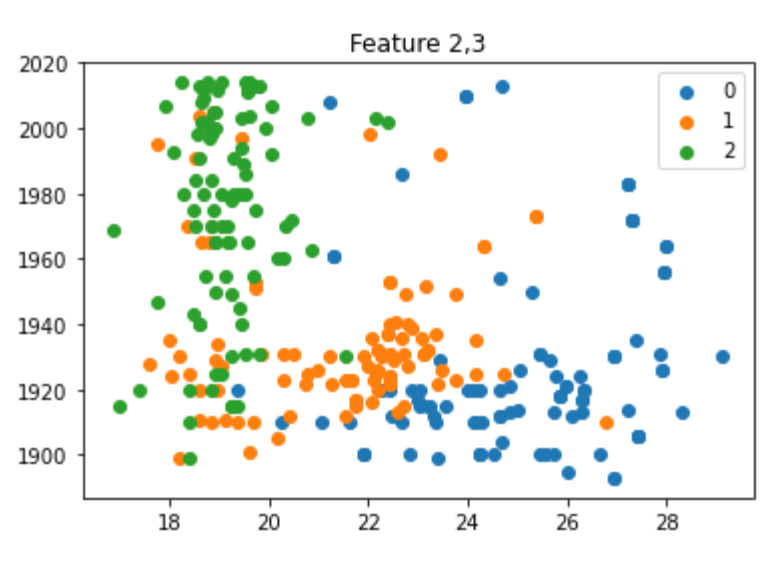
```
Out[10]: Text(0.5, 1.0, 'Feature 1,3')
```



plot (2,3)

```
In [11]:
plt.scatter(nych_0[:,1],nych_0[:,2])
plt.scatter(nych_1[:,1],nych_1[:,2])
plt.scatter(nych_2[:,1],nych_2[:,2])
plt.legend([0,1,2])
plt.title('Feature 2,3')
```

```
Out[11]: Text(0.5, 1.0, 'Feature 2,3')
```



problem2.

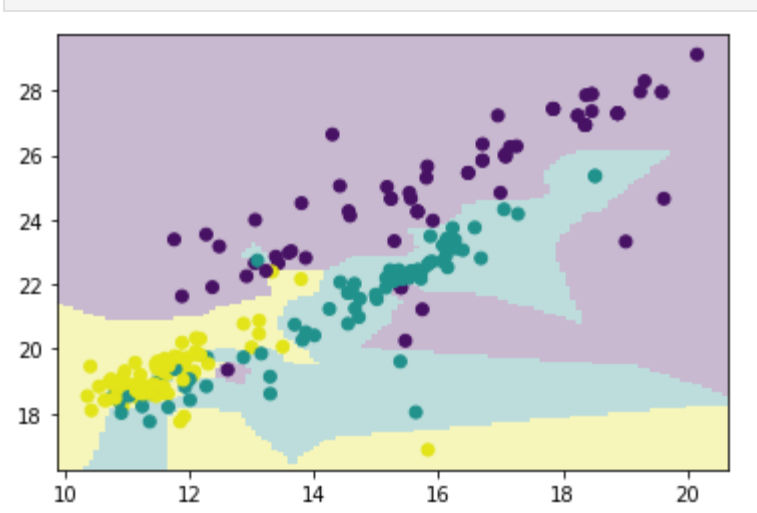
```
In [6]:
import mltools as ml
```

```
In [7]:
np.random.seed(0)
X,Y = ml.shuffleData(X,Y)
Xtr,Xva,Ytr,Yva = ml.splitData(X,Y,0.75)
```

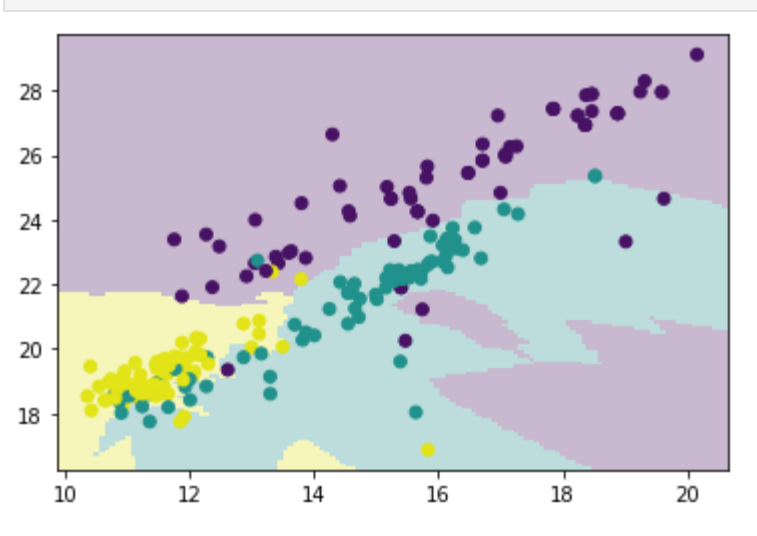
```
In [8]:
knn = ml.knn.knnClassify()
knn.train(Xtr,Ytr,1)
YvaHat = knn.predict(Xva)
```

1.

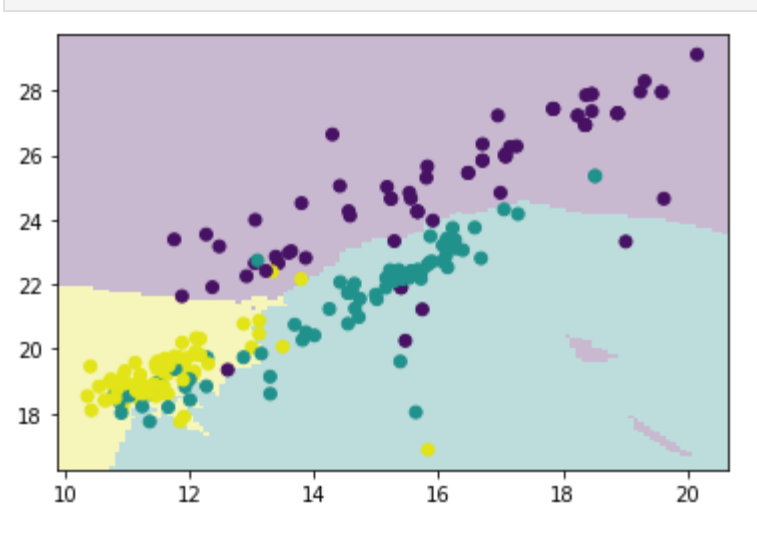
```
In [10]:
np.random.seed(0)
Y = nych[:,-1]
X = nych[:,0:2]
X,Y = ml.shuffleData(X,Y)
Xtr,Xva,Ytr,Yva = ml.splitData(X,Y,0.75)
knn = ml.knn.knnClassify()
knn.train(Xtr,Ytr,1)
YvaHat = knn.predict(Xva)
ml.plotClassify2D(knn,Xtr,Ytr)
```



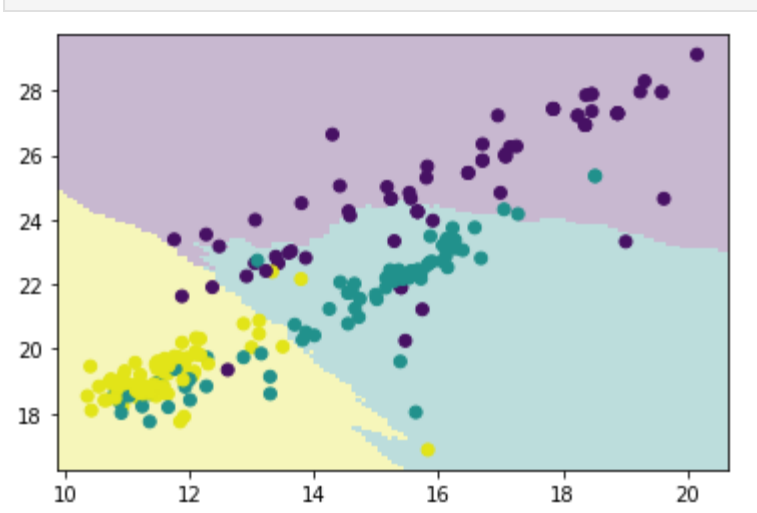
```
In [11]:
knn.train(Xtr,Ytr,5)
YvaHat = knn.predict(Xva)
ml.plotClassify2D(knn,Xtr,Ytr)
```



```
In [13]:
knn.train(Xtr,Ytr,10)
YvaHat = knn.predict(Xva)
ml.plotClassify2D(knn,Xtr,Ytr)
```



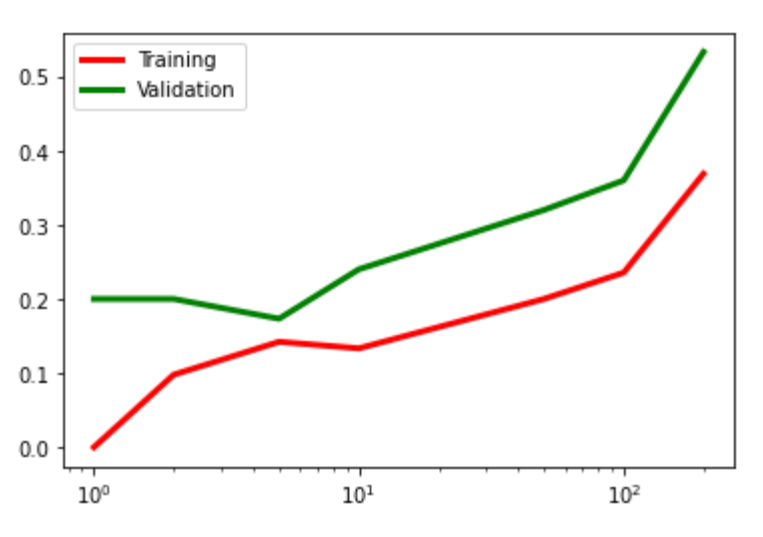
```
In [14]:
knn.train(Xtr,Ytr,50)
YvaHat = knn.predict(Xva)
ml.plotClassify2D(knn,Xtr,Ytr)
```



2.

```
In [110...
K = [1,2,5,10,50,100,200]
Y = nych[:,-1]
X = nych[:,0:2]
X,Y = ml.shuffleData(X,Y)
Xtr,Xva,Ytr,Yva = ml.splitData(X,Y,0.75)
errTrain = [None]*len(K)
errValidation = [None]*len(K)
for i,k in enumerate(K):
    knn = ml.knn.knnClassify()
    knn.train(Xtr,Ytr,k)
    YvaHat = knn.predict(Xva)
    YtrHat = knn.predict(Xtr)
    errTrain[i] = (len(Ytr) - sum(YtrHat == Ytr))/len(Ytr)
    errValidation[i] = (len(YvaHat) - sum(YvaHat == Yva))/len(YvaHat)
plt.semilogx(K, errTrain, 'r-', lw=3, label='Training')
plt.semilogx(K, errValidation, 'g-', lw=3, label='Validation')
plt.legend()
```

```
Out[110... <matplotlib.legend.Legend at 0x213d43128e0>
```

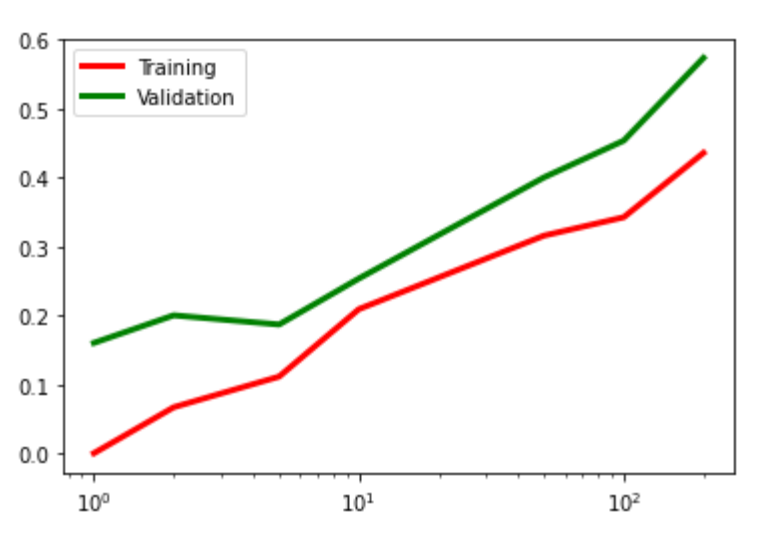


I would recommend take k = 5.

3.

```
In [115...
K = [1,2,5,10,50,100,200]
Y = nych[:,-1]
X = nych[:,0:3]
X,Y = ml.shuffleData(X,Y)
Xtr,Xva,Ytr,Yva = ml.splitData(X,Y,0.75)
errTrain = [None]*len(K)
errValidation = [None]*len(K)
for i,k in enumerate(K):
    knn = ml.knn.knnClassify()
    knn.train(Xtr,Ytr,k)
    YvaHat = knn.predict(Xva)
    YtrHat = knn.predict(Xtr)
    errTrain[i] = (len(Ytr) - sum(YtrHat == Ytr))/len(Ytr)
    errValidation[i] = (len(YvaHat) - sum(YvaHat == Yva))/len(YvaHat)
plt.semilogx(K, errTrain, 'r-', lw=3, label='Training')
plt.semilogx(K, errValidation, 'g-', lw=3, label='Validation')
plt.legend()
```

```
Out[115... <matplotlib.legend.Legend at 0x213d9e744f0>
```



The best value for k did not change, and the plots for are not very different.

Problem3

1. P(y=-1) = 0.6 and P(y = 1) = 0.4

1. P(x1 = 0|y=-1) = 3/4, P(x1 = 0 |y= 1) = 1/4, P(x1 = 1|y = -1) = 1/2, and P(x1 = 1 |y = 1) = 1/2 P(x2 = 0|y=-1) = 1/5, P(x2 = 0 |y= 1) = 4/5, P(x2 = 1|y = -1) = 1, and P(x2 = 1 |y = 1) = 0 P(x3 = 0|y=-1) = 2/3, P(x3 = 0 |y= 1) = 1/3, P(x3 = 1|y = -1) = 3/7, and P(x3 = 1 |y = 1) = 4/7 P(x4 = 0|y=-1) = 1/3, P(x4 = 0 |y= 1) = 2/3, P(x4 = 1|y = -1) = 2/7, and P(x4 = 1 |y = 1) = 5/7 P(x5 = 0|y=-1) = 4/7, P(x5 = 0 |y= 1) = 3/7, P(x5 = 1|y = -1) = 2/3, and P(x5 = 1 |y = 1) = 1/3

2. x = (0 0 0 0 0) then P(y=1) = 1/44/51.32/33/7 = 2/105 then y = -1 x = (1 1 0 1 0) then P(y=1) = 1/201/35/73/7 = 0 then y = -1

3. P(y=1 | x = (0 0 0 0 0)) = 2/105 P(y=1 | x = (1 1 0 1 0)) = 0

1. Beacuse if we use the joint probability, there are 2^5 = 32 possible combinations of x and we don't have enough data to obation the probability for that

5.I don't think we need to re-train the model. Instead, since we assume each features are independent of others, we can still use the conditional probability for x2-x5 to decide a given case.

Problem 4

1.

```
In [7]:
Y = nych[:,-1]
X = nych[:,0:2]
X,Y = ml.shuffleData(X,Y)
Xtr,Xva,Ytr,Yva = ml.splitData(X,Y,0.75)
```

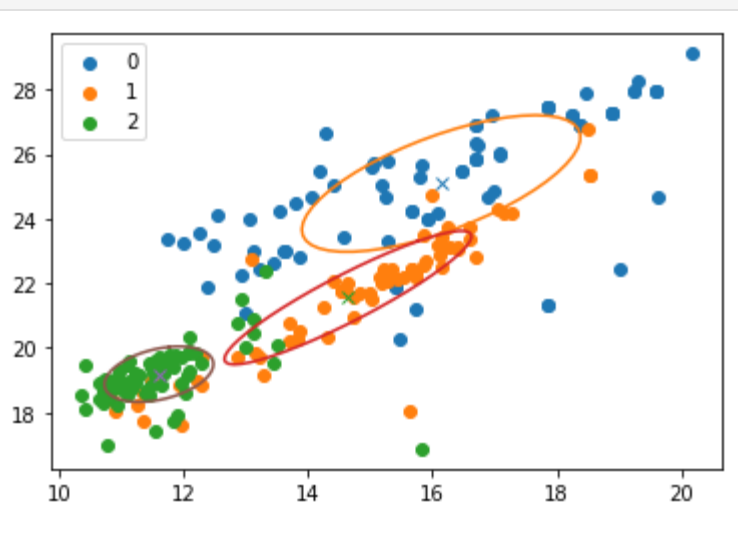
```
In [17]:
Xtr_0 = Xtr[Ytr == 0]
Xtr_1 = Xtr[Ytr == 1]
Xtr_2 = Xtr[Ytr == 2]
```

```
In [18]:
mean_vector_0 = [np.mean(Xtr_0[:,0]), np.mean(Xtr_0[:,1])]
mean_vector_1 = [np.mean(Xtr_1[:,0]), np.mean(Xtr_1[:,1])]
mean_vector_2 = [np.mean(Xtr_2[:,0]), np.mean(Xtr_2[:,1])]
```

```
In [21]:
cov_matrix_0 = np.cov(Xtr_0[:,0],Xtr_0[:,1])
cov_matrix_1 = np.cov(Xtr_1[:,0],Xtr_1[:,1])
cov_matrix_2 = np.cov(Xtr_2[:,0],Xtr_2[:,1])
```

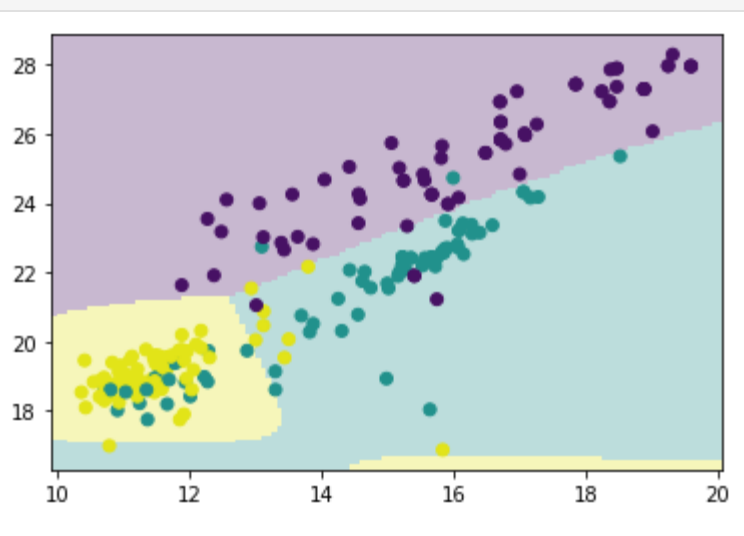
2.

```
In [23]:
plt.scatter(Xtr_0[:,0],Xtr_0[:,1])
plt.scatter(Xtr_1[:,0],Xtr_1[:,1])
plt.scatter(Xtr_2[:,0],Xtr_2[:,1])
plt.legend([0, 1, 2])
ml.plotGauss2D(mean_vector_0,cov_matrix_0)
ml.plotGauss2D(mean_vector_1,cov_matrix_1)
ml.plotGauss2D(mean_vector_2,cov_matrix_2)
```



3.

```
In [80]:
bc = ml.bayes.gaussClassify(Xtr,Ytr)
ml.plotClassify2D(bc, Xtr, Ytr)
```



```
In [93]:
Yhat = bc.predict(Xva)
error_rate = (len(Yhat)-sum(Yhat == Yva))/len(Yhat)
error_rate
```

```
Out[93]: 0.17333333333333334
```

Problem5

I did this homework all by myself. Zhengran Ji.