Name: Hootan Hosseinzadeganbushehri

[redacted]

UCI Email: hootanh@uci.edu

Problem 1 - Part 1:

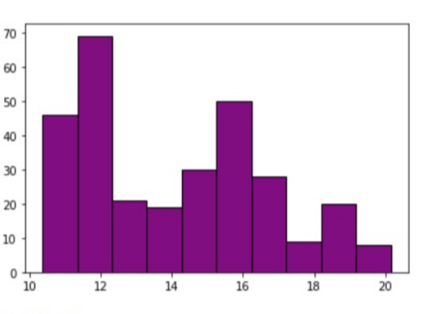
In [161]:
```
import numpy as np
import matplotlib.pyplot as plt

nych = np.genfromtxt(r"data\nyc_housing.txt", delimiter = None)
Y = nych[:,-1]
X = nych[:,0:-1]

print("\033[1m" + "\nNumber of data points:", X.shape[0], "\033[1m" + "\nNumber of features:", X.shape[1])
```

```
Number of data points: 300
Number of features: 3
```

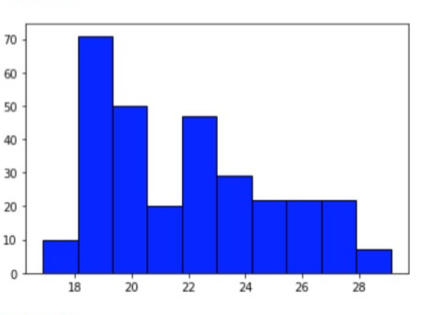
Problem 1 - Part 2:

In [162]:
```
print("\033[1m" + "\nFeature 1:")
plt.hist(X[:,0], color = "purple", edgecolor = "black")
plt.show()
print()
print("\033[1m" + "Feature 2:")
plt.hist(X[:,1], color = "blue", edgecolor = "black")
plt.show()
print()
print("\033[1m" + "Feature 3:")
plt.hist(X[:,2], color = "gray", edgecolor = "black")
plt.show()
```

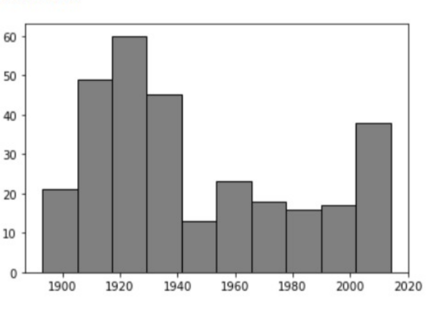
Feature 1:



Feature 2:



Feature 3:



Problem 1 - Part 3:

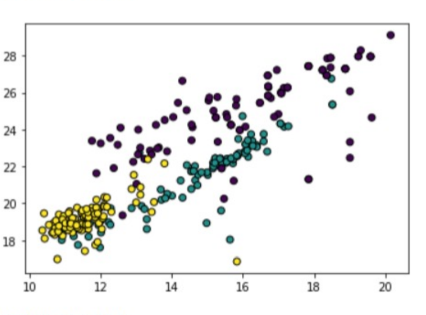
In [163]:
```
print("\033[1m" + "\nFeature 1:")
print("Mean:", np.mean(X[:,0]))
print("Standard deviation:", np.std(X[:,0]))
print()
print("\033[1m" + "Feature 2:")
print("Mean:", np.mean(X[:,1]))
print("Standard deviation:", np.std(X[:,1]))
print()
print("\033[1m" + "Feature 3:")
print("Mean:", np.mean(X[:,2]))
print("Standard deviation:", np.std(X[:,2]))
```

```
Feature 1:
Mean: 14.118392438424483
Standard deviation: 2.569090284260317

Feature 2:
Mean: 21.907116176170856
Standard deviation: 2.9785784999947165

Feature 3:
Mean: 1946.3533333333332
Standard deviation: 35.39889577687731
```

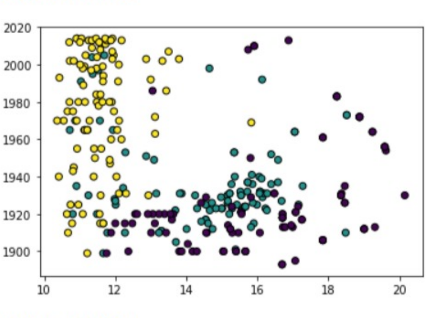
Problem 1 - Part 4:

In [164]:
```
print("\033[1m" + "\nFeature 1 and 2:")
plt.scatter(X[:,0], X[:,1], c = Y, edgecolors = "black")
plt.show()
print()
print("\033[1m" + "Feature 1 and 3:")
plt.scatter(X[:,0], X[:,2], c = Y, edgecolors = "black")
plt.show()
print()
print("\033[1m" + "Feature 2 and 3:")
plt.scatter(X[:,1], X[:,2], c = Y, edgecolors = "black")
plt.show()
```

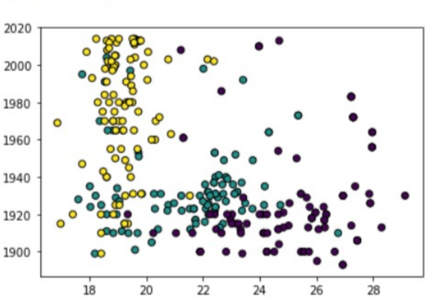
Feature 1 and 2:



Feature 1 and 3:



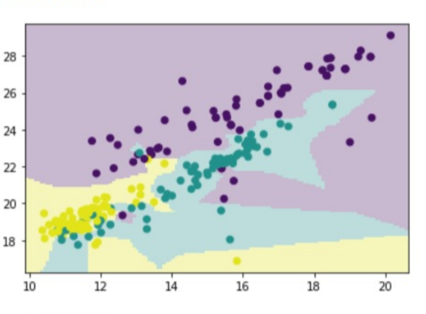
Feature 2 and 3:



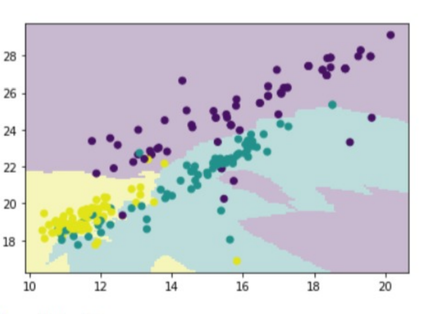
Problem 2 - Part 1:

In [165]:
```
nych = np.genfromtxt(r"data\nyc_housing.txt", delimiter = None)
Y = nych[:,-1]
X = nych[:,0:-1]

import mltools as ml

np.random.seed(0)
X, Y = ml.shuffleData(X, Y)

Xtr, Xva, Ytr, Yva = ml.splitData(X, Y, 0.75)
k = [1, 5, 10, 50]

for i in range(len(k)):
    print()
    print("\033[1m" + "When K is " + str(k[i]) + ":")
    knn = ml.knn.knnClassify()
    knn.train(Xtr[:, 0:2], Ytr, k[i])
    ml.plotClassify2D(knn, Xtr[:, 0:2], Ytr)
    plt.show()
```

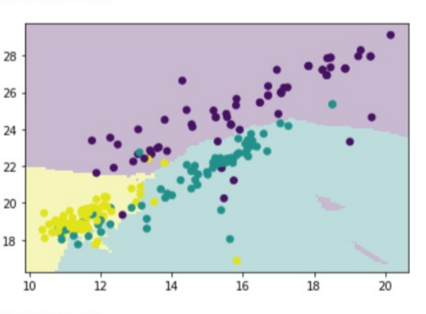
When K is 1:



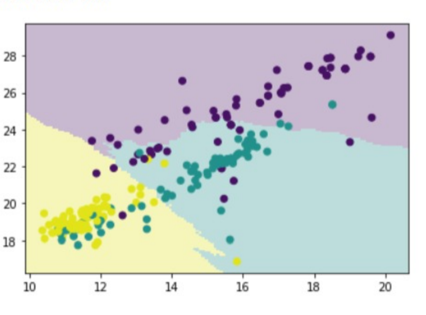
When K is 5:



When K is 10:



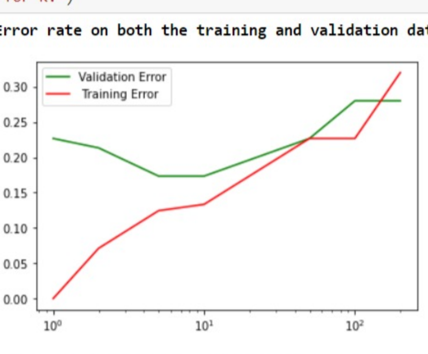
When K is 50:



Problem 2 - Part 2:

In [166]:
```
K = [1, 2, 5, 10, 50, 100, 200]
errTrain = [None]*len(K)
errValid = [None]*len(K)

for i, k in enumerate(K):
    learner = ml.knn.knnClassify(Xtr[:,0:2], Ytr, k)
    errTrain[i] = learner.err(Xtr[:,:2], Ytr)
    errValid[i] = learner.err(Xva[:,:2], Yva)

plt.semilogx(K, errValid, "Green", label = "Validation Error")
plt.semilogx(K, errTrain, "Red", label = " Training Error")
plt.legend()
print()
print("\033[1m" + "Error rate on both the training and validation data:")
plt.show()
print("From the figure above, the best choice is k =", K[np.argmin(errValid)], "so I would recommend", K[np.argmin(errValid)],
"for K.")
```

Error rate on both the training and validation data:




```
From the figure above, the best choice is k = 5 so I would recommend 5 for K.
```

Problem 2 - Part 3:

In [167]:
```
K = [1, 2, 5, 10, 50, 100, 200]
errTrain = [None]*len(K)
errValid = [None]*len(K)

for i, k in enumerate(K):
    learner = ml.knn.knnClassify(Xtr, Ytr, k)
    errTrain[i] = learner.err(Xtr, Ytr)
    errValid[i] = learner.err(Xva, Yva)

plt.semilogx(K, errValid, "Green", label = "Validation Error")
plt.semilogx(K, errTrain, "Red", label = " Training Error")
plt.legend()
print()
print("\033[1m" + "Error rate on both the training and validation data:")
plt.show()
print("From the figure above, the best choice is k =", K[np.argmin(errValid)], "so I would recommend", K[np.argmin(errValid)],
"for K.")
```

Error rate on both the training and validation data:


```
From the figure above, the best choice is k = 1 so I would recommend 1 for K.
```

Yes, the plots are different and also my recommendation for the best K is different as well; therefore, a change in the number of the features can change both plots and the best K.

Problem 3 - Part 1:

p(y=-1) = 6/10 = 3/5

p(y=1) = 4/10 = 2/5

p(x1=1|y=-1) = 3/6 = 1/2

p(x2=1|y=-1) = 5/6

p(x3=1|y=-1) = 4/6 = 2/3

p(x4=1|y=-1) = 5/6

p(x5=1|y=-1) = 2/6 = 1/3

p(x1=1|y=1) = 3/4

p(x2=1|y=1) = 0/4 = 0

p(x3=1|y=1) = 3/4

p(x4=1|y=1) = 2/4 = 1/2

p(x5=1|y=1) = 1/4

p(x1=0|y=-1) = 3/6 = 1/2

p(x2=0|y=-1) = 1/6

p(x3=0|y=-1) = 2/6 = 1/3

p(x4=0|y=-1) = 1/6

p(x5=0|y=-1) = 4/6 = 2/3

p(x1=0|y=1) = 1/4

p(x2=0|y=1) = 4/4 = 1

p(x3=0|y=1) = 1/4

p(x4=0|y=1) = 2/4 = 1/2

p(x5=0|y=1) = 3/4

Problem 3 - Part 2:

p(00000|y=1) = (4/10)*(1/4)*(1)*(1/4)*(2/4)*(3/4) = 0.009375

p(00000|y=-1) = (6/10)*(3/6)*(1/6)*(2/6)*(1/6)*(4/6) = 0.001851852

==>

p(00000|y=1) > p(00000|y=-1) ==> Class y = 1 is bigger than class y = -1

==> Therefore, the email must be read.

---

p(11010|y=1) = (4/10)*(3/4)*(0)*(3/4)*(2/4)*(1/4) = 0

p(11010|y=-1) = (6/10)*(3/6)*(5/6)*(2/6)*(5/6)*(4/6) = 0.046296296

==>

p(11010|y=1) < p(11010|y=-1) ==> Class y = -1 is bigger than class y = 1

==> Therefore, the email must be discarded.

Problem 3 - Part 3:

p(y=1|00000) =

= p(00000|y=1) p(y=1) / p(y=1) p(00000|y=1) + p(00000|y=-1) p(y=-1)

= 0.8351

==> p(y=1|00000) = 0.8351

---

p(y=1|11010) =

= p(11010|y=1) p(y=1) / p(y=1) p(11010|y=1) + p(11010|y=-1) p(y=-1)

= 0

==> p(y=1|11010) = 0

Problem 3 - Part 4:

Since here we have many parameters, joint Bayes classifier is going to take a lot of time and can be very slow as well. Also, joint Bayes classifier can increase the complexity in the computation. However, it is more efficient for us to use naive Bayes classifier instead because the data features are independent and can be done by using fewer parameters. In addition, naive Bayes classifier can simplify the calculations for the probabilities and increase the speed of the process which is going to be faster than joint Bayes classifier. Therefore, we should not use a joint Bayes classifier for these dara.

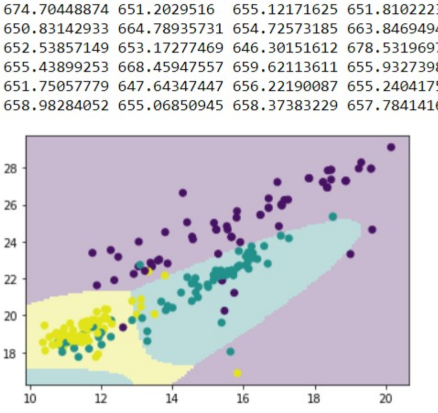
Problem 3 - Part 5:

We do not need to re-train the model because all the features are independent from each other; therefore, it is not needed to re-train the model even after deleting a feature. Furthermore, we simply just exclude the first feature(x1) meaning that we do not use p(x1=1|y=1) which is the probability associated with the first feature in the calculation and there is no change in the rest of the probabilities. Therefore, we do not have to re-train the model or re-calculating parameters after excluding the probability for the first feature(x1).

Problem 4:

In [171]:
```
nych = np.genfromtxt(r"data\nyc_housing.txt", delimiter = None)
Y = nych[:,-1]
X = nych[:,0:-1]

import mltools as ml

np.random.seed(0)
X, Y = ml.shuffleData(X, Y)

Xtr, Xva, Ytr, Yva = ml.splitData(X, Y, 0.75)

learner = ml.knn.knnClassify(Xtr[:,0:2], Ytr, k)
errTrain = learner.err(Xtr[:,:2], Ytr)
errValid = learner.err(Xva[:,:2], Yva)

Xtr = Xtr[:,:2]
Xtr_class0 = Xtr[Ytr==0]
Xtr_class1 = Xtr[Ytr==1]

data = np.array([Xtr_class0,Y])

print(Xtr.shape)

print(np.mean(Xtr, axis = 0))

bc = ml.bayes.gaussClassify(Xtr[:, 0:2], Ytr);
ml.plotClassify2D(bc, Xtr[:, 0:2], Ytr);
Xtr_class0.transpose()

print(errTrain)
print(errValid)
print(X[Y==0].mean(axis=0))
print(X[Y==1].mean(axis=1))
a=np.mean(Xtr_class0)
b=np.cov(Xtr_class0)

#ml.plotGauss2D(a, b)
```

```
(225, 2)
[14.02464449 21.7962618 ]
0.32
0.28
[  16.14898626   25.07251957 1926.94      ]
[651.09655676 665.25450304 655.68381395 661.67278942 655.03270718
 654.03664401 659.32921024 645.33296736 649.8053967  658.18405506
 656.79896687 650.45539944 653.15506344 655.12973532 654.64209185
 655.16339453 650.14932592 652.14840455 675.98900382 659.26977693
 656.59400079 656.22190087 657.0954563  668.45947557 653.32652193
 666.80405228 647.63714081 658.81938777 657.0954563  678.22398977
 661.20386612 651.09655676 673.52646224 652.68764132 649.43251925
 648.80905793 666.41368164 655.49807526 647.0515745  655.89236648
 661.61235966 654.45248877 656.57314004 672.29174325 656.76956238
 653.88246328 663.58940357 642.95272036 658.48430021 677.98685989
 654.03664401 653.44343742 652.63851768 655.93273989 654.67057218
 653.90265585 656.63136018 647.81288981 663.11482739 652.55861412
 653.18423227 652.46943826 647.03947521 653.4636694  654.32569645
 677.18452869 650.47478687 658.37383229 672.29174325 653.22479355
 674.70448874 651.2029516  655.12171625 651.81022234 654.19691872
 650.83142933 664.78935731 654.72573185 663.84694944 653.32652193
 652.53857149 653.17277469 646.30151612 678.53196973 663.58940357
 655.43899253 668.45947557 659.62113611 655.93273989 654.30420021
 651.75057779 647.64347447 656.22190087 655.24041756 652.53111556
 658.98284052 655.06850945 658.37383229 657.7841416  659.80317764]
```



Problem5:

I did HW1 completely on my own by using Piazza threads before I start to do my homework. In addition, I used the lecture notes and read through the base codes which was provided in the zip file. I completely followed the academic honesty guidelines which is on our canvas website and I did not discuss my homework with anyone in-person.