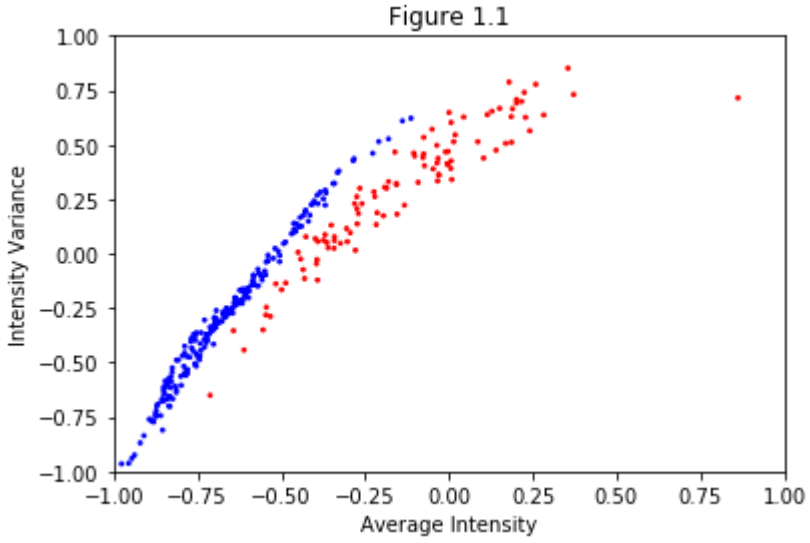
In [1]:
```
#From the console, run the following
#pip install numpy
#pip install scipy
#pip install scikit-learn
#pip install matplotlib

# Import required packages here (after they are installed)
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as mp
from pylab import show

from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.ensemble import AdaBoostClassifier


from statistics import mean, stdev, median, mode
```

In [14]:
```
def minimum(x,y):
    min = np.argmin(y)

    return x[min]
```

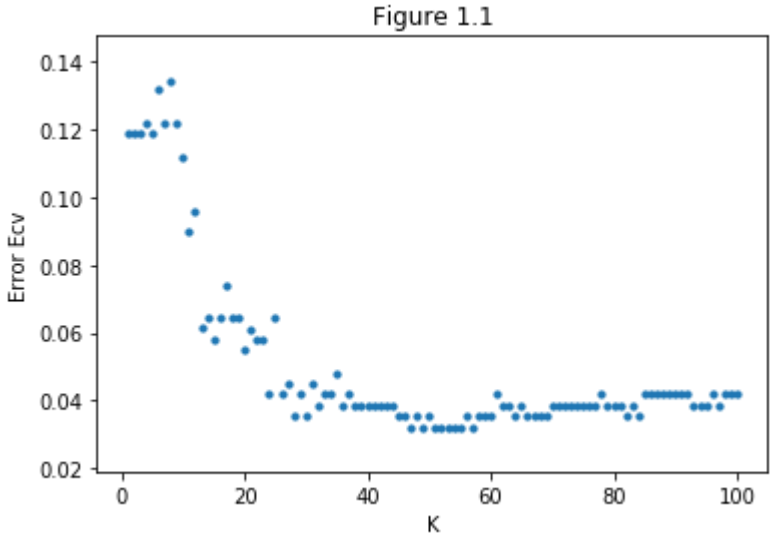
In [2]:
```
# Load data. csv file should be in the same folder as the notebook for this to
work, otherwise
# give data path.
data = np.loadtxt("data.csv")
```

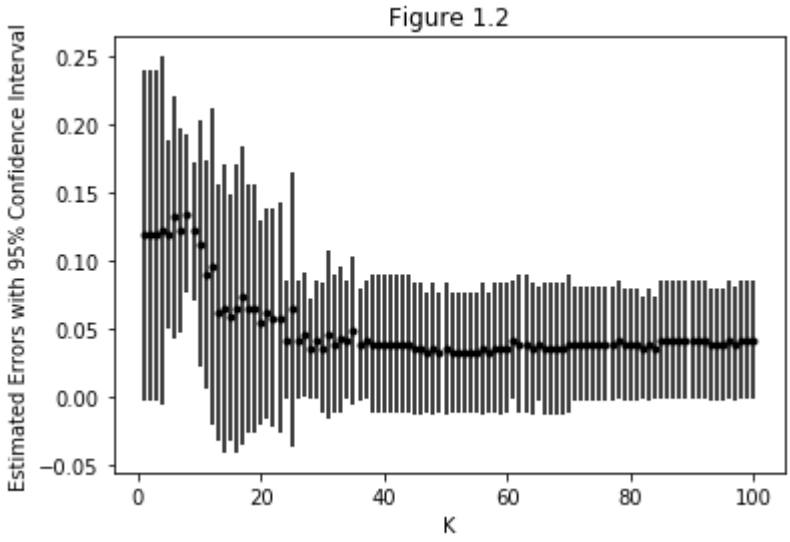
In [3]:
```
#shuffle the data and select training and test data
np.random.seed(100)
np.random.shuffle(data)


features = []
digits = []


for row in data:
    #import the data and select only the 1's and 5's
    if(row[0]==1 or row[0]==5):
        features.append(row[1:])
        digits.append(str(row[0]))


#Select the proportion of data to use for training.
#Notice that we have set aside 80% of the data for testing
numTrain = int(len(features)*.2)

trainFeatures = features[:numTrain]
testFeatures = features[numTrain:]
trainDigits = digits[:numTrain]
testDigits = digits[numTrain:]
```

In [4]:
```
#Convert the 256D data (trainFeatures) to 2D data
#We need X and Y for plotting and simpleTrain for building the model.
#They contain the same points in a different arrangement

X = []
Y = []
simpleTrain = []

#Colors will be passed to the graphing library to color the points.
#1's are blue: "b" and 5's are red: "r"
colors = []
for index in range(len(trainFeatures)):
    #produce the 2D dataset for graphing/training and scale the data so it is
 in the [-1,1] square
    xNew = 2*np.average(trainFeatures[index])+.75
    yNew = 3*np.var(trainFeatures[index])-1.5
    X.append(xNew)
    Y.append(yNew)
    simpleTrain.append([xNew,yNew])
    #trainDigits will still be the value we try to classify. Here it is the st
ring "1.0" or "5.0"
    if(trainDigits[index]=="1.0"):
        colors.append("b")
    else:
        colors.append("r")

#plot the data points
### https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html
mp.scatter(X,Y,s=3,c=colors)

#specify the axes
mp.xlim(-1,1)
mp.xlabel("Average Intensity")
mp.ylim(-1,1)
mp.ylabel("Intensity Variance")
mp.title("Figure 1.1")
#display the current graph
show()
```



In [19]:
```
# USING 2D dimensional data
x = []
y = []
z = []
p =[]
m=[]
std =[]
for i in range(1,101):
    #print(i)
    model  = AdaBoostClassifier(n_estimators = i)
    #model2.predict(testFeatures)
    cvs = cross_val_score(model, simpleTrain, trainDigits, cv = 10, scoring='a
ccuracy')
    err = 1-cvs
    evsm = err.mean()
    temp = stdev(err)
    temp2 = evsm + temp
    m.append(evsm)
    std.append(2*temp)
    p.append(temp2)
    x.append(i)
    y.append(evsm)
    z.append([x,evsm])

# print(len(x))
# print(len(y))
# print(count)
mp.scatter(x,y, s=10)
mp.xlabel("K")
mp.ylabel("Error Ecv")
mp.title("Figure 1.1")
mp.show()


mp.errorbar(x, m, yerr=std, fmt='.k');

mp.xlabel("K")

mp.ylabel("Estimated Errors with 95% Confidence Interval")
mp.title("Figure 1.2")
show()
```





In [23]:
```
b = minimum(x,m)
b
```

Out[23]: 47

Considering 95 % confident Interval

In [22]:
```
b = minimum(x,p)
b
```

Out[22]: 28

Considering error mean and 95 % confidence interval, The optimal number of estimators is 28