Data Mining and Machine Learning

LECTURE 10:
Supervised classification: k-nn classifiers

Dr. Edgar Acuna
Department of Mathematics

University of Puerto Rico- Mayaguez

academic.uprm.edu/eacuna

The k nearest neighbors method (Fix y Hodges, 1951) is used to estimate the density function of a random variable.

In the supervised classification context the k-nn method is used to estimate the class conditional density $f(\mathbf{x}/C_j)$, of the predictors $\mathbf{x}$ in a given class C_j.

k-nn is a nonparametric method since none assumption on the distribution of the predictors is made.

K-nn univariate density estimation

- Let x_1, x_2,… ..x_n be a random sample fron a unknown density function f(x), and let t a real number where f is going to be estimated.
- Recall that the probability of x lies in the interval (t-h,t+h), can be approximated by 2hf(t), where f is the density function and h is a constant near to zero.

  On the other hand, such probability can also be estimated by k/n, where k is the number of observations in the interval (t-h,t+h). In k-nn estimation, k is pre-fixed and h is computed according to h.

  Formally, let d(x,y)=|x-y| be the usual distance between the points x and y of the real line.

Suppose that we have computed all the distances $d(x_i,t)=|x_i-t|$ and that these are ordered in a increasing way, such as:

$d_1(t) \leq d_2(t) \leq \ldots \ldots . \leq d_n(t)$

- The the estimador of the density function at the point t using k nearest neighbors is given by:

$$\hat{f}(t) = \frac{k}{2nd_k(t)}$$

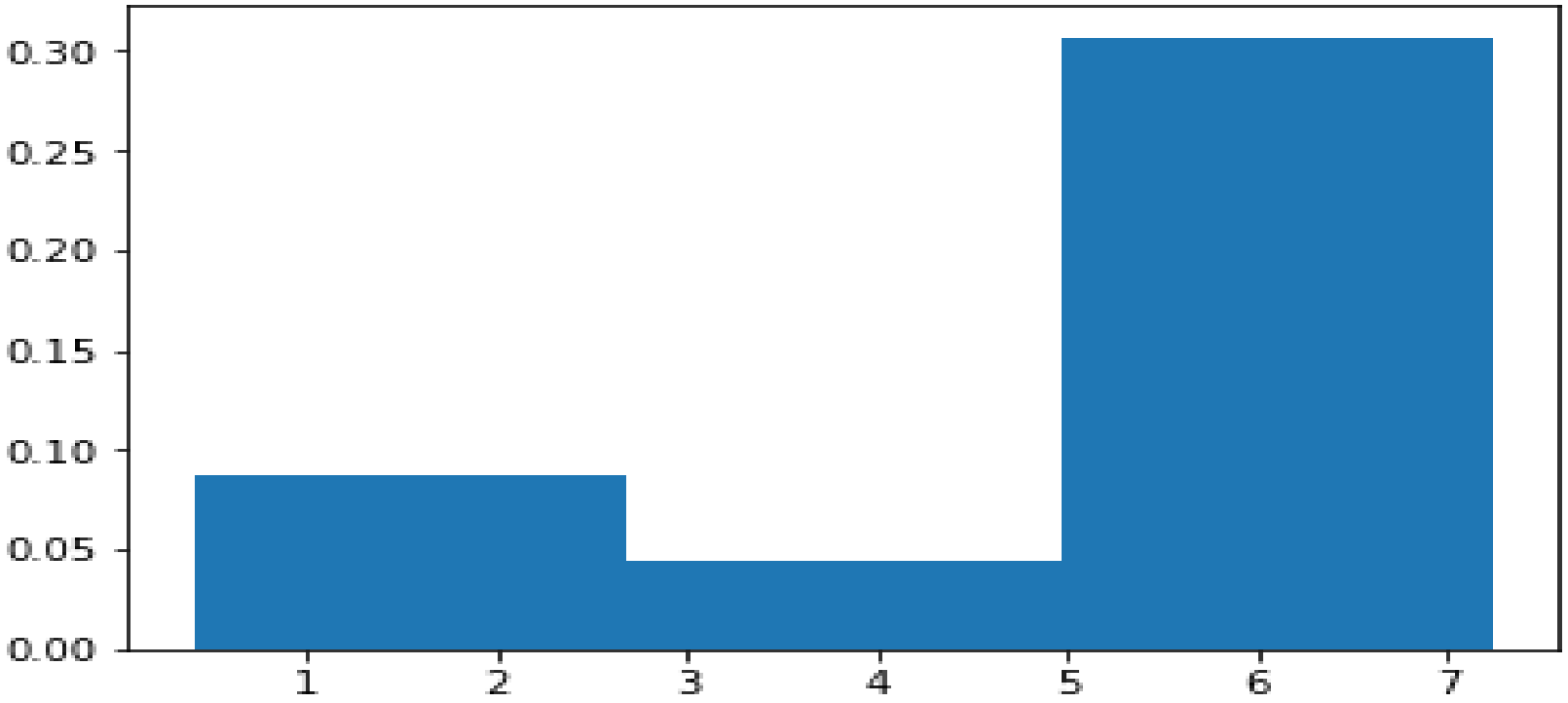
Example: Density estimation

Given the following sample of size n=10, estimate the its density function

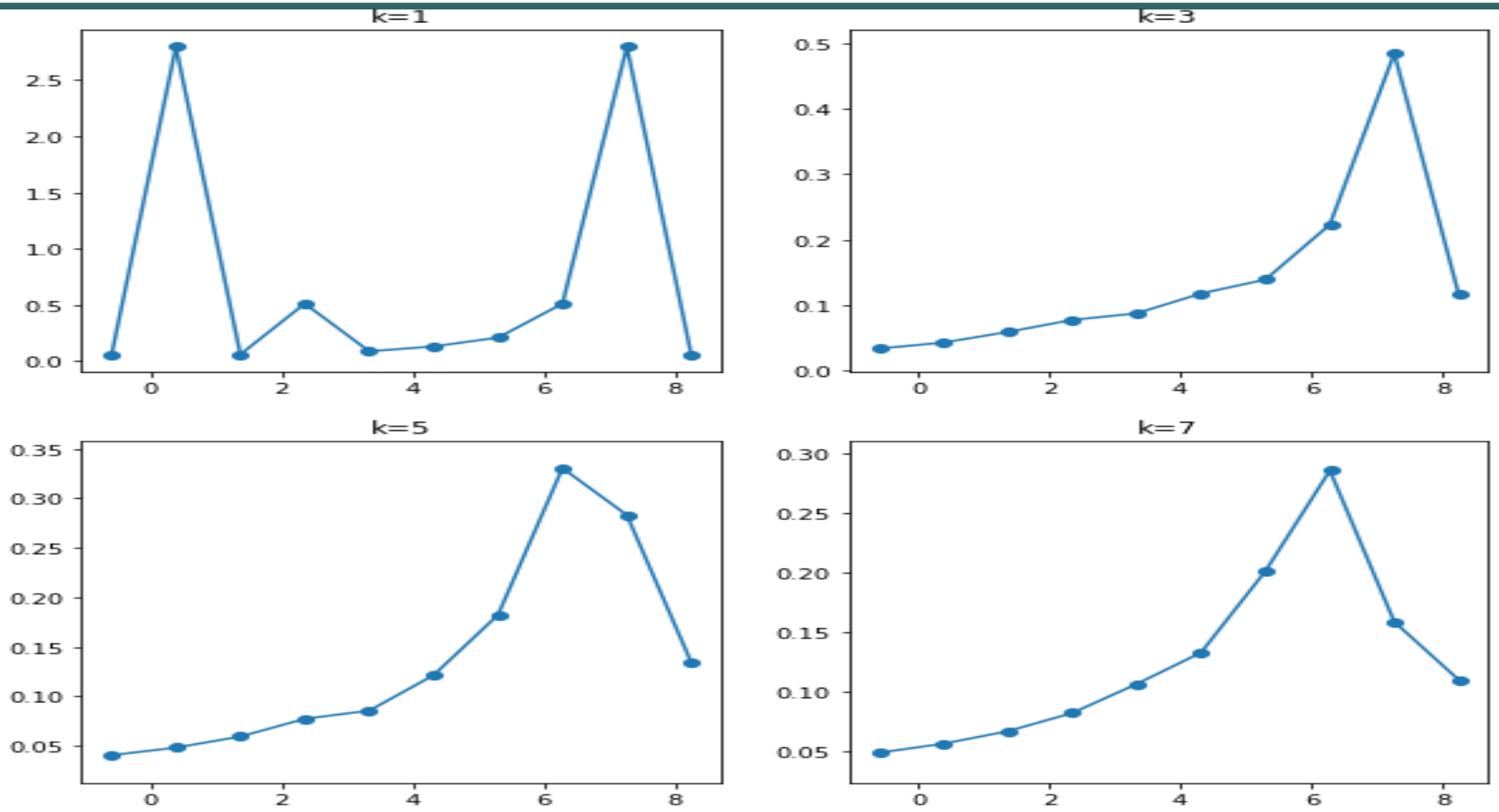
7.03860 6.38018 6.95461 2.25521 7.24608 6.62930 3.92734 0.40701 5.59448 5.05751

The normalized histogram (area = 1) is shown in the next slide.
Noticet that this density estimator is not continuous.

Density estimation using histogram



Knn density estimation for several values of k



Multivariate k-nn density estimation

- The estimate of the density function is given by

$$\hat{f}(\mathrm{x}) = \frac{k}{n v_k(\mathrm{x})}$$

- where $v_k(\mathbf{x})$ is the volume of the ellipsoid centered at **x** and with radius $r_k(\mathbf{x})$, which is the distance of x to the k-th nearest point.

The k-nn classifier

In supervised classification he k-nn algorithm is very easy to appy(Cover and Hart, 1967)

In fact, if the if the conditional density function ***f(x/C$_i$)*** *of class* Ci that appears in the posterior probability expression

$$P(C_i / \mathrm{x}) = \frac{f(\mathrm{x} / C_i)\pi_i}{f(\mathrm{x})}$$

Is estimated using k-nn, then to classify a object, with measurements given by the vector x , into class Ci , it has to hold

$$\frac{k_i \pi_i}{n_i v_k(\mathrm{x})} > \frac{k_j \pi_j}{n_j v_k(x)}$$

for j≠I, where k_i y k_j are the k neighbors in classes C_i and C_j respectively.

The k-nn classifier (cont)

Assuming proportional priors to the size of the classes (ni/n and nj/n respectively) the last equation is equivalent to:

$$k_i > k_j \text{ para } j \neq i$$

Then, the classification of an object **x** will be as it follows:
1) Find the k objects that are closer in distance to the object **x**, k usually is and odd number: 1, 3 etc. to avoid ties.
2) If the majority of these k objects belong to class Ci, then object x is assigned to that class.
In case of a tie, the class is assigned randomly.

The k-nn classifier (cont)

- The k-nn method has two problems; the distance to be chosen and the election of the number of neighbors k.
- The most elementary distance that can be chosen is the Euclidean distance given by d(**x**,**y**)=(**x**-**y**)'(**x**-**y**). This distance may cause problems when the predictors have very different scale of measurement. It is a good idea to normalize the data before applying k-nn. Other distance that can be used is the Manhattan distance given by d(**x**,**y**)=|**x**-**y**|.
- Enas and Choi (1996) using simulation perform an experiment to determine the optimal value of k for the two class problem and they found $k=n^{3/8}$ when the covariance matrices of the two groups are similar and $k=n^{2/8}$ when the covariance matrices of the two groups are very different. In here, n= number of instances.

The k-nn classifier (cont)

- The bias of the misclassification error increases as k increases, but the variance decreases.
- Cover and Hart (1967) showed that the miclassification error rate of a k-nn classifier is at most 2 times the optimal error rate (error of a Bayesian classifer where the posterior are explicitly known).

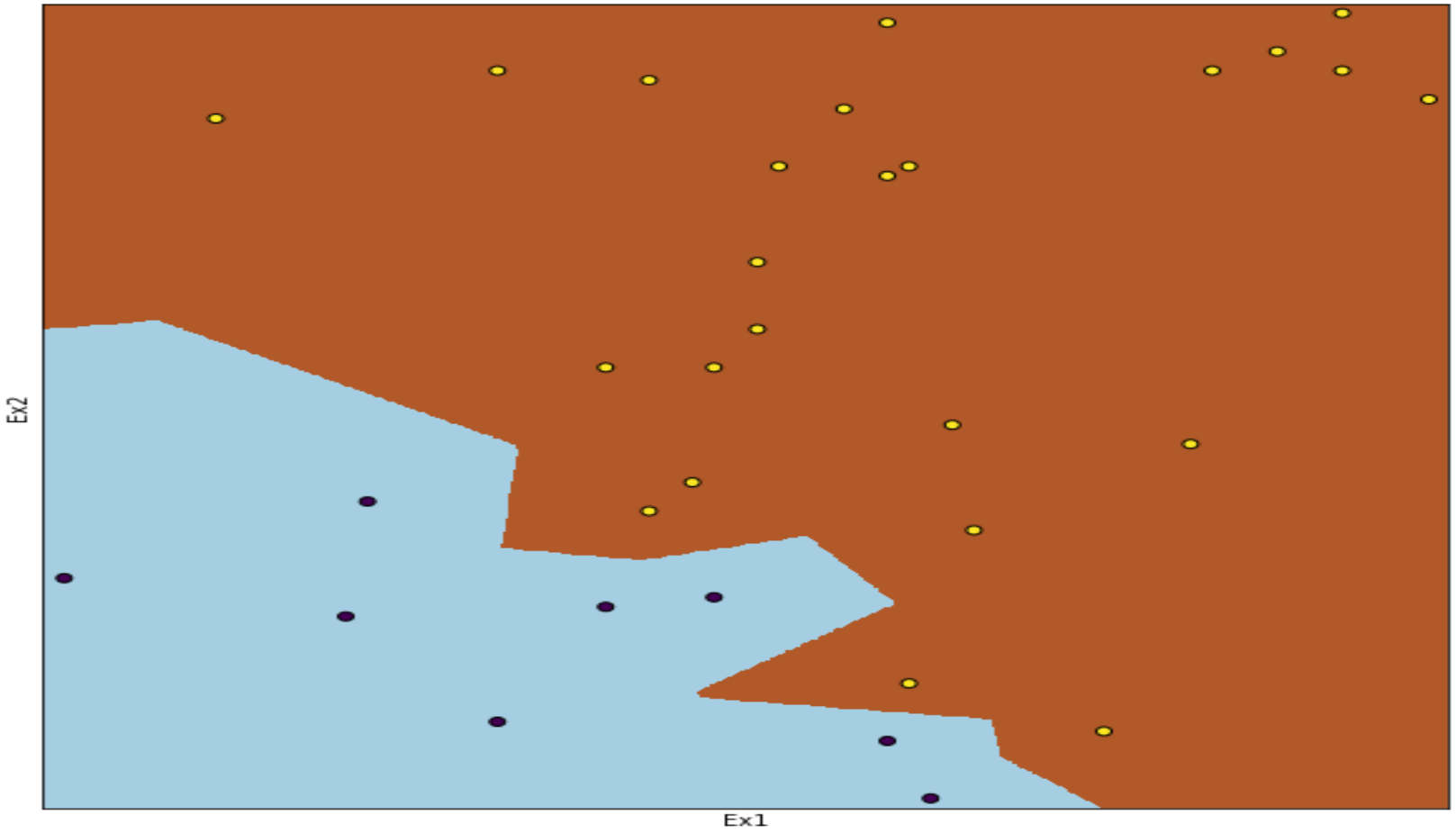
Example: Predicting final grade

```
df=pd.read_csv("c://PW-PR/eje1dis.csv")
#Extracting the predictor matrix and the class column
y=df['Nota']
X=df.iloc[:,0:2]
#creating a numerical class column "pass"
lb_make = LabelEncoder()
df["pass"] = lb_make.fit_transform(df["Nota"])
#Tambien se puede usar y.as_matrix() para la clasificacion
y1=df["pass"].as_matrix()
X1=X.as_matrix()
#Constructing the classifier
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X1, y1)
```

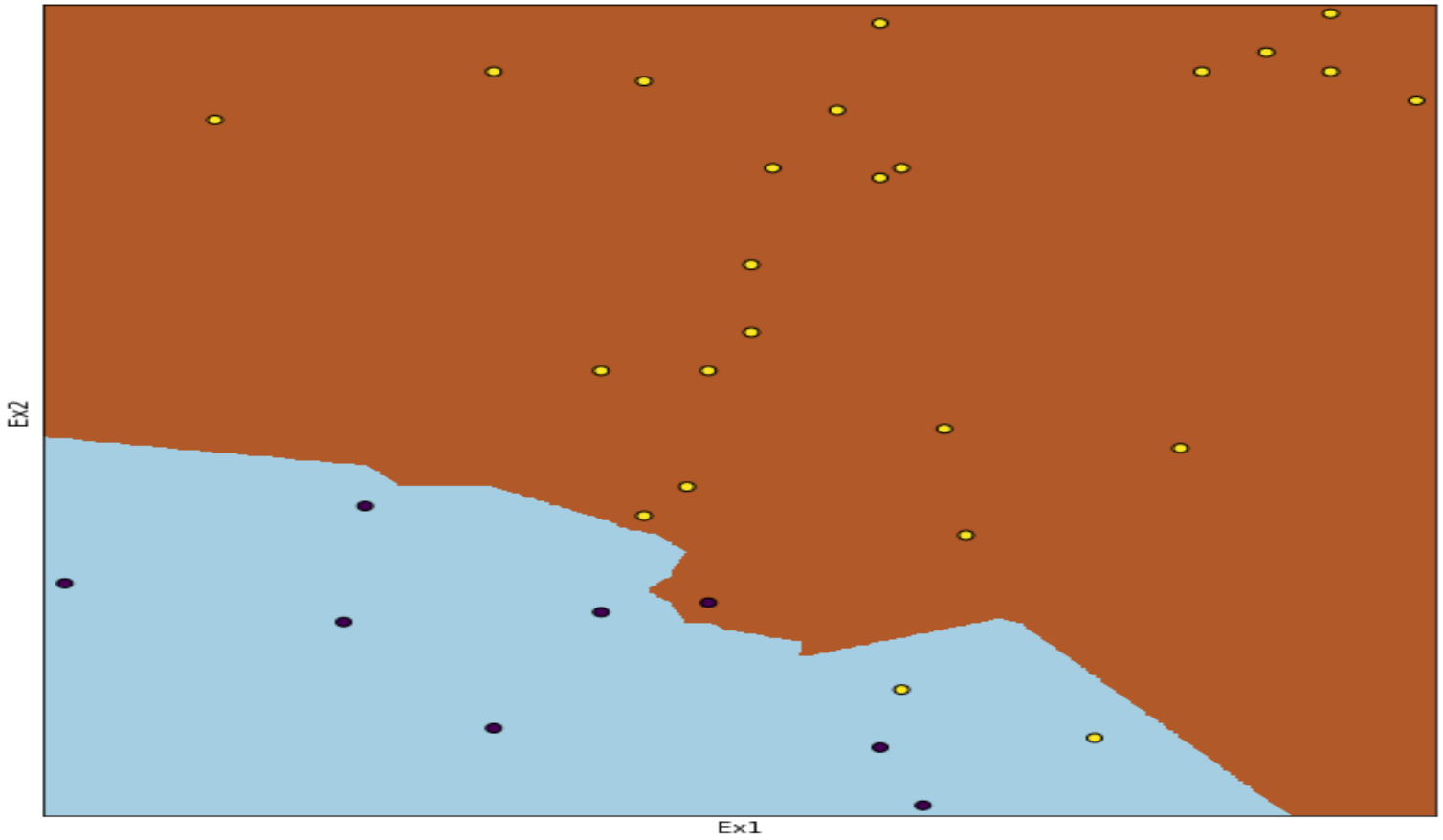
Example: Predicting final grade (cont)

```
#Making the predictions
pred=neigh.predict(X1)
print predprint pred
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0]
#Calculating the number of errors
error=(y!=pred).sum()
print “This is the number of errors=", error
This is the number of errors= 0
```

Example: Decision boundary(k=1)



Example: Decision Boundary (k=7)



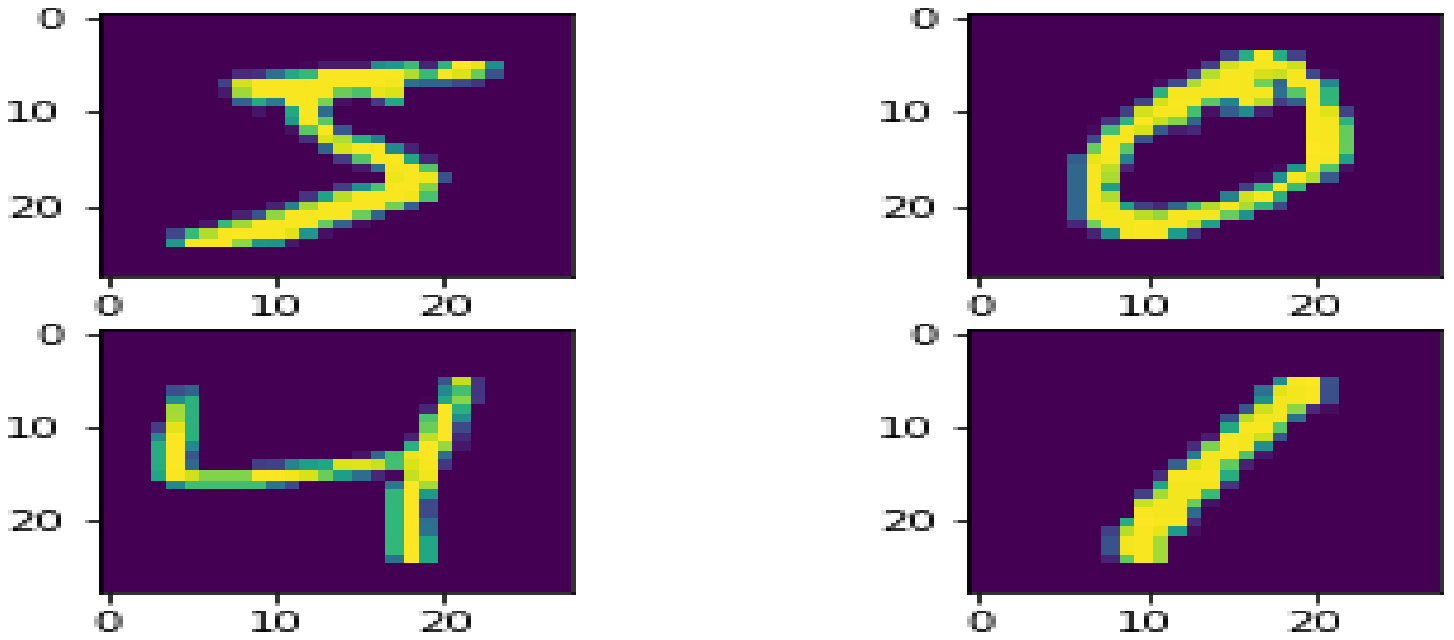
Example: Diabetes

```
url= "http://academic.uprm.edu/eacuna/diabetes.dat"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = pd.read_table(url, names=names)
y=data['class']
X=data.iloc[:,0:8]
y1=y.as_matrix()
X1=X.as_matrix()
#Estimacion de la precision usando k=5 vecinos usando validacion cruzada
from sklearn.model_selection import cross_val_score
neigh = KNeighborsClassifier(n_neighbors=5)
scores = cross_val_score(neigh, X1, y1, cv=10)
scores
ray([ 0.67532468, 0.79220779, 0.71428571, 0.67532468, 0.66233766, 0.74025974, 0.7012987 , 0.79220779,
      0.71052632, 0.75 ])
print("Accuracy using k=5 neighbors: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
Accuracy using k=5 neighbors: 0.72 (+/- 0.09)
```

Applying k-nn to digit recognition

The MNIST dataset was used by LeCun, Cortes y Burges, The training dataset consists of 60k images of the digits 0 to 9 and the test dataset consists of 10k images. Each row of the datasets has 785 entries containing 784 pixels (28x28) and the first entry is the digit label.
The datasets formatted as csv are available at https://pjreddie.com/ The training dataset is 104MB and the test is 17MB.
Scikit-learn has a very small subset of the MNIST dataset
Visualizing the digits:

Visualizing some rows of MNIST



Applying Knn to a subset of MNIST

It takes more than 10 minutes to work with the full dataset, therefore we will use an small subset available in scikit-learn

```
# load el MNIST digits dataset
mnist = datasets.load_digits()
print ("the number of records is:",len(mnist.data))
the number of records is: 1797
# Training and testing split,
# 75% for training and 25% for testing
(trainData, testData, trainLabels, testLabels) = train_test_split(np.array(mnist.data), mnist.target,
test_size=0.25, random_state=42)
# Showing the size of each data split
print("training data points: {}".format(len(trainLabels)))
print("testing data points: {}".format(len(testLabels)))
```

Applying Knn to a subset of MNIST (cont)

```
# Applying  k-nn with k-5
model = KNeighborsClassifier(n_neighbors=5)
model.fit(trainData, trainLabels)

# Predicting the tesData labels
predictions = model.predict(testData)

# Evaluating the performance of the classifier for each digit
print("EVALUATION ON TESTING DATA")
print(classification_report(testLabels, predictions))
```

Applying Knn to a subset of MNIST (cont)

```
# Applying  k-nn with k-5
model = KNeighborsClassifier(n_neighbors=5)
model.fit(trainData, trainLabels)

# Predicting the tesData labels
predictions = model.predict(testData)

# Evaluating the performance of the classifier for each digit
print("EVALUATION ON TESTING DATA")
print(classification_report(testLabels, predictions))
```

Applying Knn to a subset of MNIST (cont)

EVALUATION ON TESTING DATA
precision recall f1-score support
0 1.00 1.00 1.00 43
1 1.00 1.00 1.00 37
2 1.00 1.00 1.00 38
3 1.00 1.00 1.00 46
4 0.98 1.00 0.99 55
5 0.98 0.98 0.98 59
6 1.00 1.00 1.00 45
7 1.00 1.00 1.00 41
8 1.00 1.00 1.00 38
9 0.98 0.96 0.97 48
Most of the digits are predicted with 100% of accuracy

Other nonparametric classifier

- Based on kernel density estimation(mixtures of continuous densities). But theirs disadvantage is that they are computationally expensive. The most used is the Gaussian Kernel.
- In general, a nonparametric method suffers of the curse of the dimensionality problem. The number of instances must grow almost exponentially to the number of predictor to have a good estimation.