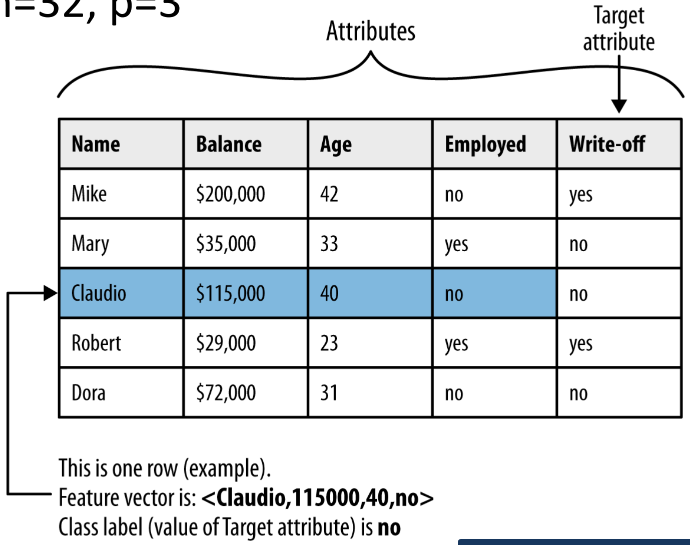
Alberi di classificazione

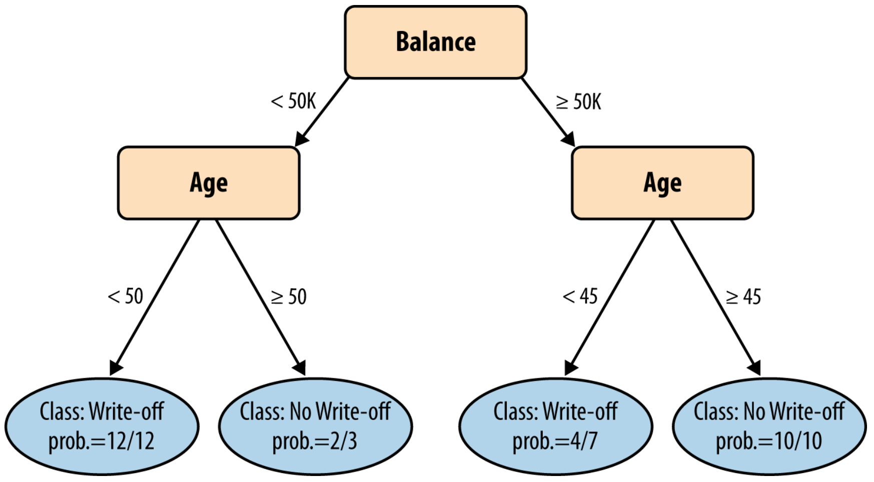
n=32, p=3


Attributes
Target attribute

Name	Balance	Age	Employed	Write-off
Mike	$200,000	42	no	yes
Mary	$35,000	33	yes	no
Claudio	$115,000	40	no	no
Robert	$29,000	23	yes	yes
Dora	$72,000	31	no	no

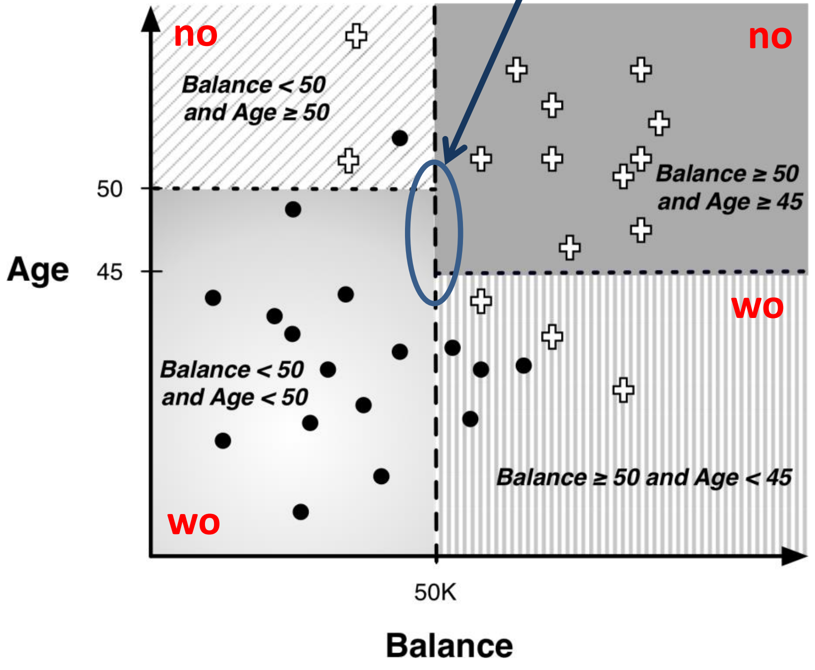
This is one row (example).
Feature vector is: **<Claudio,115000,40,no>**
Class label (value of Target attribute) is **no**

fitting


Balance
< 50K
≥ 50K
Age
Age
< 50
≥ 50
< 45
≥ 45
Class: Write-off prob.=12/12
Class: No Write-off prob.=2/3
Class: Write-off prob.=4/7
Class: No Write-off prob.=10/10

‘Age’ non è sufficiente a classificare, qui.

Ogni albero ha una rappresentazione grafica ed una a regole



Plotting (a valle)

i relativi decision boundary (i cut point delle varie dimensioni)

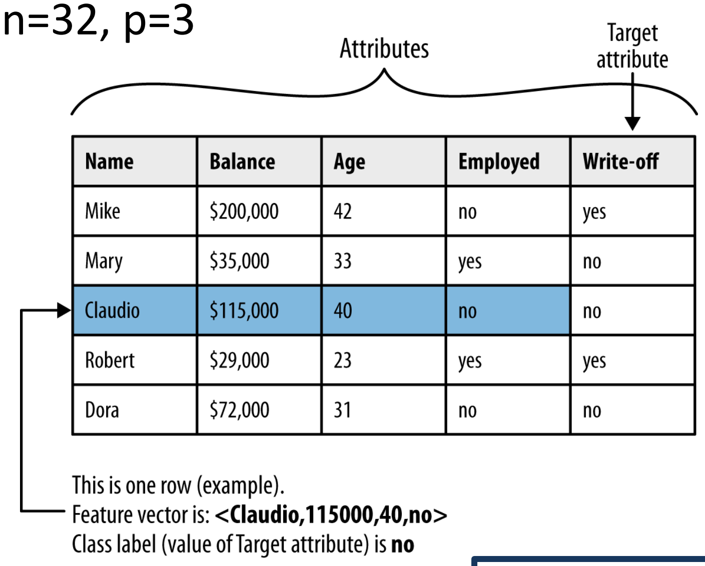
```
If Balance < 50K
    If Age < 50 then assign ‘wo’
    else assign ‘no’;
Else
    If Age < 45 then assign ‘wo’
    else assign ‘no’.
```

Animazione di alberi

Fonte: DSfB (Provost & Fawcett)

Alberi di classificazione

n=32, p=3



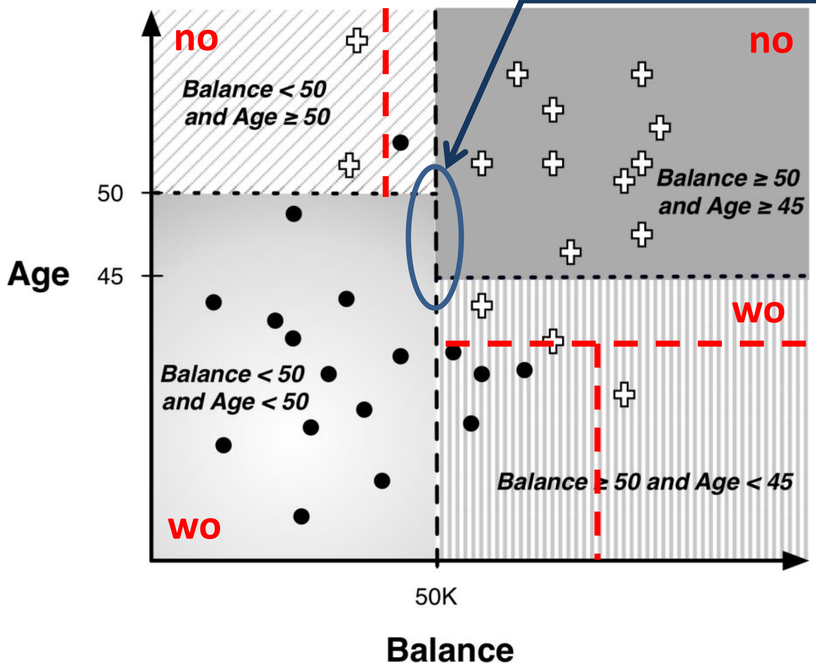
Attributes

Target attribute

Name	Balance	Age	Employed	Write-off
Mike	$200,000	42	no	yes
Mary	$35,000	33	yes	no
Claudio	$115,000	40	no	no
Robert	$29,000	23	yes	yes
Dora	$72,000	31	no	no

This is one row (example).
Feature vector is: **<Claudio,115000,40,no>**
Class label (value of Target attribute) is **no**

‘Age’ non è sufficiente a classificare, qui.



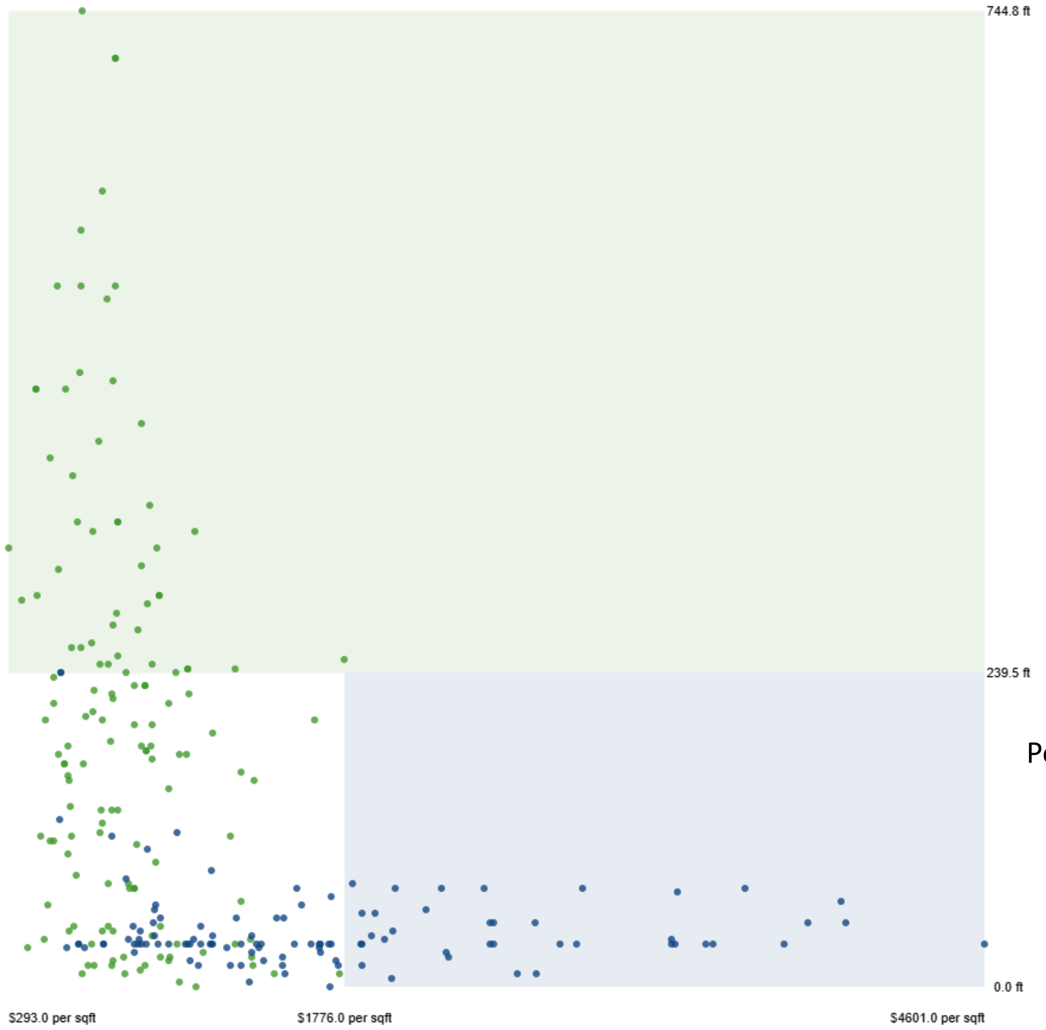
fitting

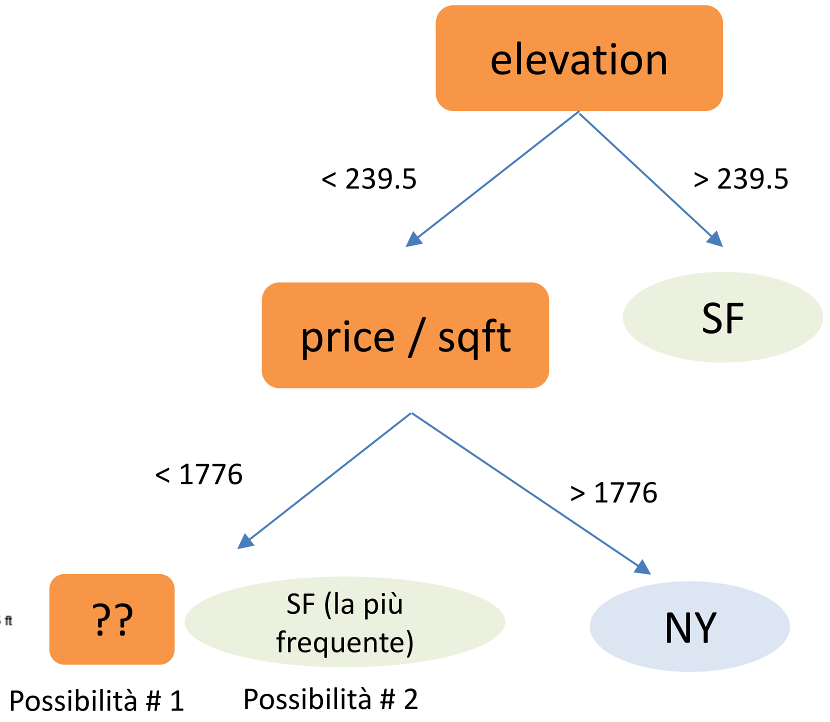
Plotting (a valle)

i relativi decision boundary (i cut point delle varie dimensioni)

```
If Balance < 50K
    If Age < 50 then assign ‘wo’
    else
        if Balance < 40 then assign ‘no’
                else assign ‘wo’;
Else
    If Age < 45 then
        if Age < 35 then assign ‘wo’
           else
              if Balance < 80 then assign ‘wo’
                 else assign ‘no’
    else assign ‘no’.
```

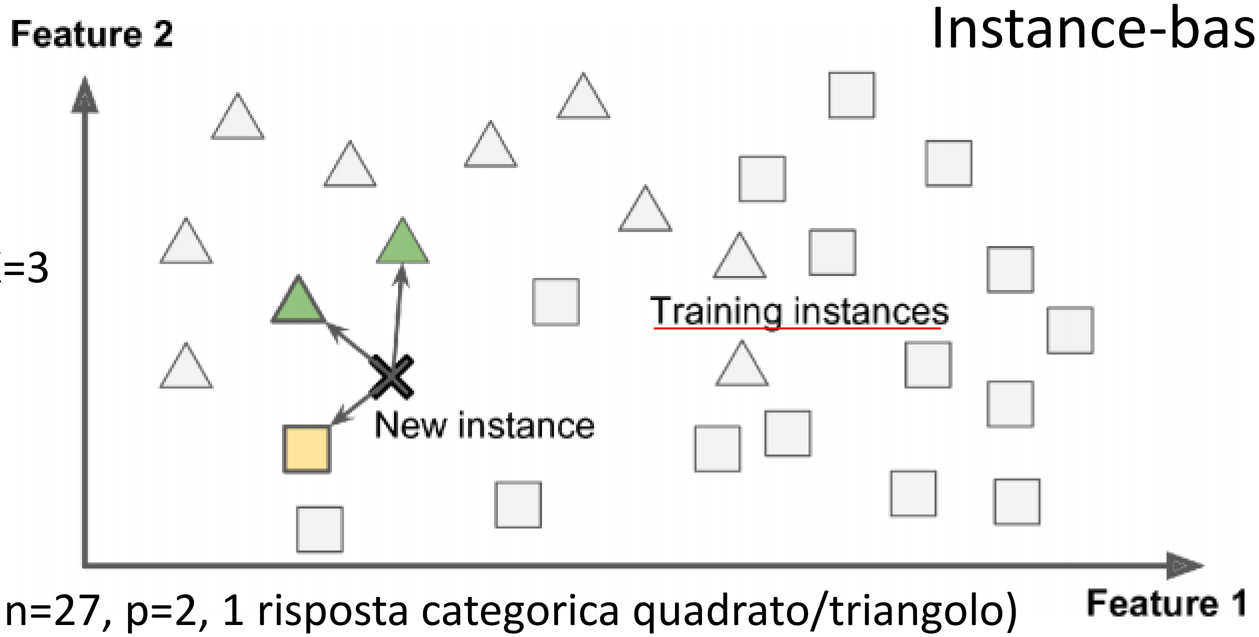
Fonte: DSfB (Provost & Fawcett)


744.8 ft
239.5 ft
0.0 ft
$293.0 per sqft
$1776.0 per sqft
$4601.0 per sqft


elevation
< 239.5
> 239.5
price / sqft
SF
< 1776
> 1776
??
SF (la più frequente)
NY
Possibilità # 1
Possibilità # 2

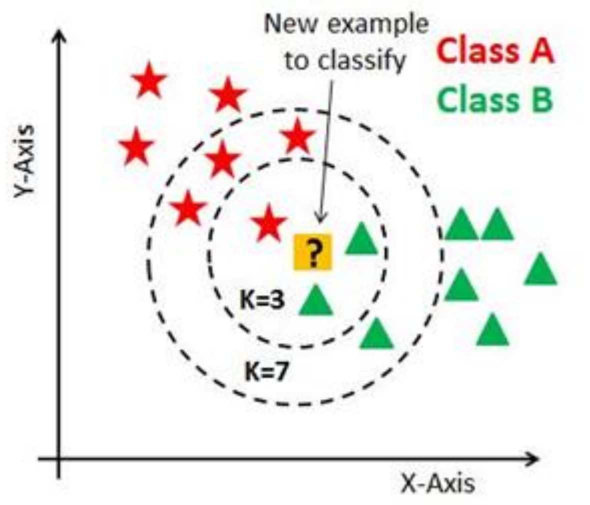
Instance-based learning (es. K-nn)

K=3



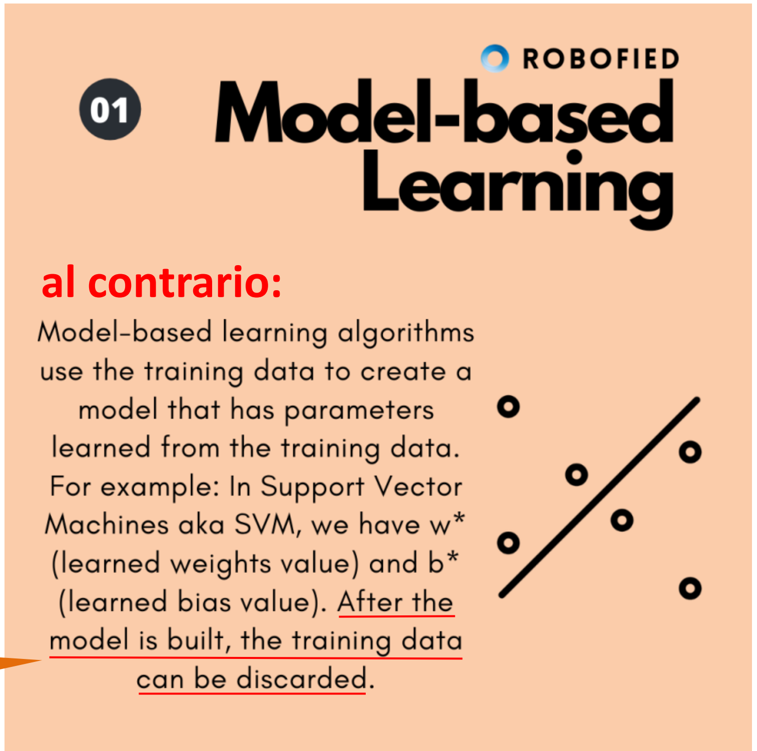
n=27, p=2, 1 risposta categorica quadrato/triangolo)

K-nn è un algoritmo *lazy* (pigro) perché rimanda il calcolo al momento in cui è nota la nuova istanza da prevedere (in altre parole, non c'è un vero training)



n=15, p=2, 1 risposta categorica stella/triangolo)

K=3 → triangolo
K=7 → stella
K=15 → triangolo


ROBOFIED

01 **Model-based Learning**

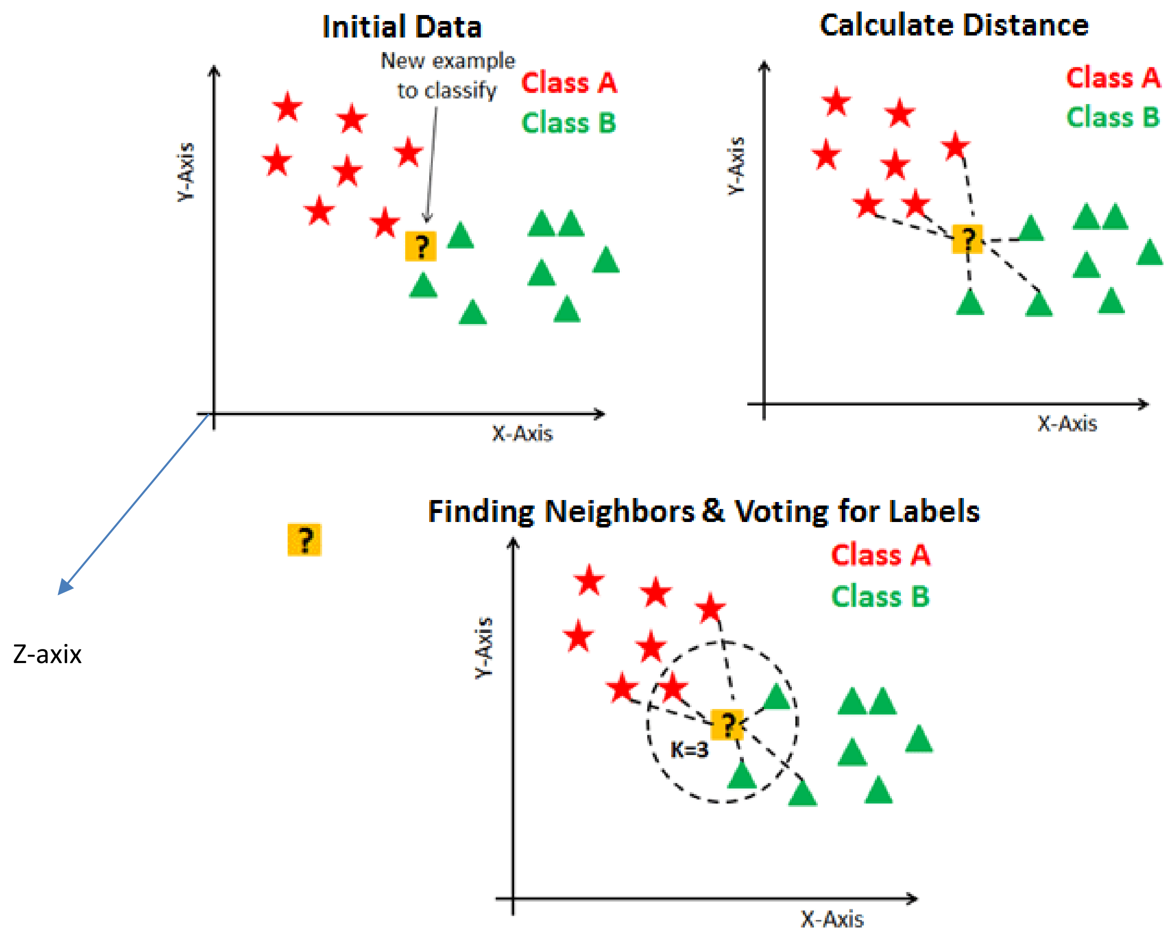
al contrario:

Model-based learning algorithms use the training data to create a model that has parameters learned from the training data. For example: In Support Vector Machines aka SVM, we have w* (learned weights value) and b* (learned bias value). After the model is built, the training data can be discarded.

Non nel K-nn!

Suppose P1 is the point, for which label needs to predict. First, you find the k closest point to P1 and then classify points by majority vote of its k neighbors. Each object votes for their class and the class with the most votes is taken as the prediction. For finding closest similar points, you find the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance and Minkowski distance. KNN has the following basic steps:

1. Calculate distance
2. Find closest neighbors
3. Vote for labels



Fonte: datacamp

Eager Vs. Lazy Learners

Eager learners mean when given training points will construct a generalized model before performing prediction on given new points to classify. You can think of such learners as being ready, active and eager to classify unobserved data points.

Lazy Learning means there is no need for learning or training of the model and all of the data points used at the time of prediction. Lazy learners wait until the last minute before classifying any data point. Lazy learner stores merely the training dataset and waits until classification needs to perform. Only when it sees the test tuple does it perform generalization to classify the tuple based on its similarity to the stored training tuples. Unlike eager learning methods, lazy learners do less work in the training phase and more work in the testing phase to make a classification. Lazy learners are also known as instance-based learners because lazy learners store the training points or instances, and all learning is based on instances.

Curse of Dimensionality

(predictors)

KNN performs better with a lower number of features than a large number of features. You can say that when the number of features increases than it requires more data. Increase in dimension also leads to the problem of overfitting. To avoid overfitting, the needed data will need to grow exponentially as you increase the number of dimensions. This problem of higher dimension is known as the Curse of Dimensionality.

To deal with the problem of the curse of dimensionality, you need to perform principal component analysis before applying any machine learning algorithm, or you can also use feature selection approach. Research has shown that in large dimension Euclidean distance is not useful anymore. Therefore, you can prefer other measures such as cosine similarity, which get decidedly less affected by high dimension.

How do you decide the number of neighbors in KNN?

Now, you understand the KNN algorithm working mechanism. At this point, the question arises that How to choose the optimal number of neighbors? And what are its effects on the classifier? The number of neighbors(K) in KNN is a hyperparameter that you need choose at the time of model building. You can think of K as a controlling variable for the prediction model.

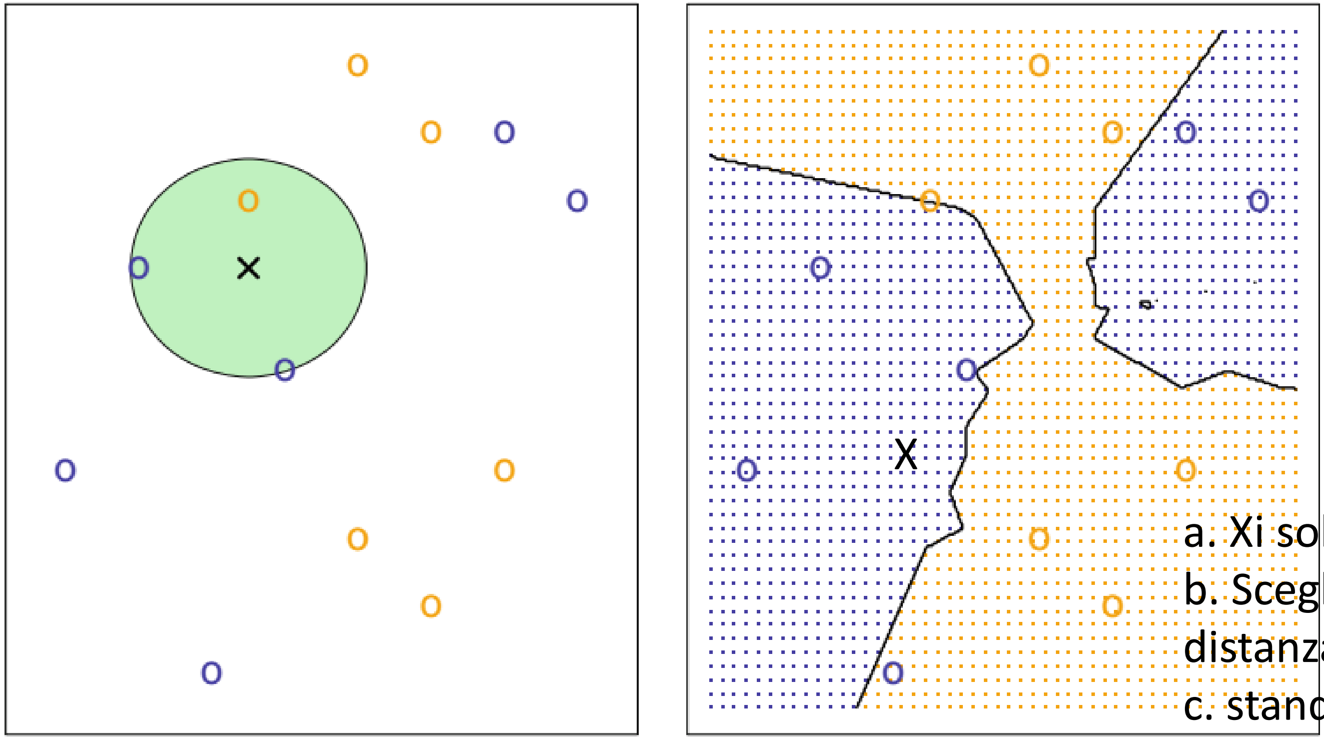
Research has shown that no optimal number of neighbors suits all kind of data sets. Each dataset has it's own requirements. In the case of a small number of neighbors, the noise will have a higher influence on the result, and a large number of neighbors make it computationally expensive. Research has also shown that a small amount of neighbors are most flexible fit which will have low bias but high variance and a large number of neighbors will have a smoother decision boundary which means lower variance but higher bias.

Generally, Data scientists choose as an odd number if the number of classes is even. You can also check by generating the model on different values of k and check their performance. You can also try Elbow method here.

Instance-based learning (es. 3-nn)

Training set: n=12 (p=2), 1 risposta binaria

X=x.1,x.2

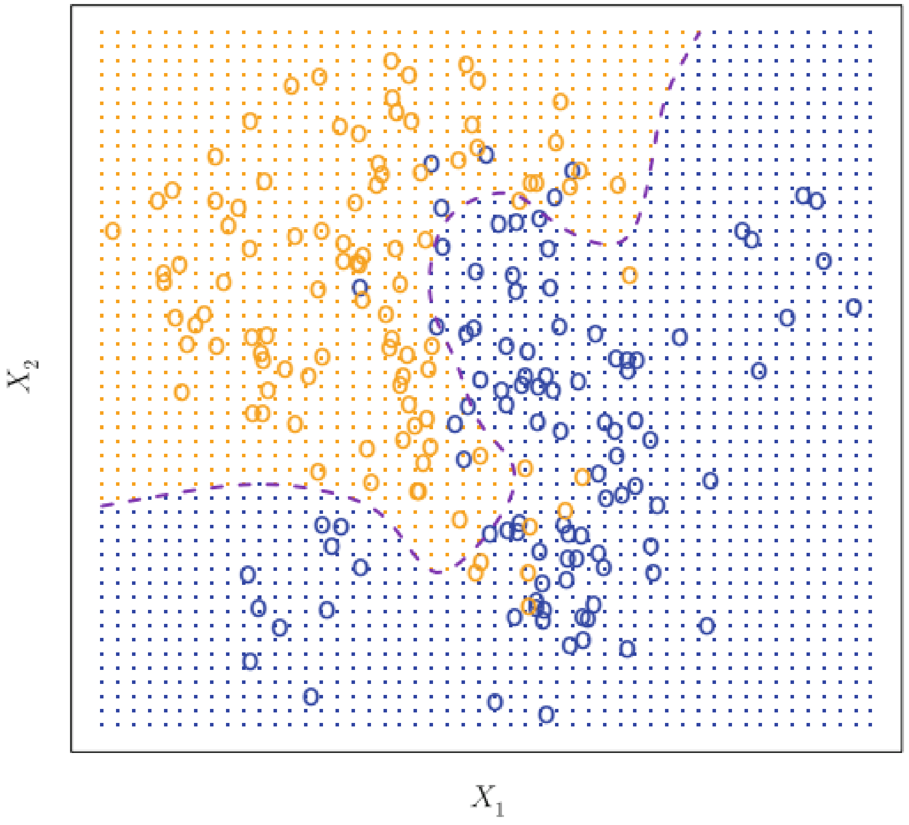


a. Xi solo numeriche?!
b. Scegliere un tipo di distanza
c. standardizzare

FIGURE 2.14. *The KNN approach, using K = 3, is illustrated in a simple situation with six blue observations and six orange observations.* Left: *a test observation at which a predicted class label is desired is shown as a black cross. The three closest points to the test observation are identified, and it is predicted that the test observation belongs to the most commonly-occurring class, in this case blue.* Right: *The KNN decision boundary for this example is shown in black. The blue grid indicates the region in which a test observation will be assigned to the blue class, and the orange grid indicates the region in which it will be assigned to the orange class.*

Fonte: ISLR di Hastie & Tibshirani

Instance-based learning (es. K-nn)



I **dati** ed il **Decision Boundary di Bayes** (noto perché dati simulati, per i quali è cioè nota la distribuzione congiunta – quasi mai).

FIGURE 2.13. *A simulated data set consisting of* 100 *observations in each of two groups, indicated in blue and in orange. The purple dashed line represents the Bayes decision boundary. The orange background grid indicates the region in which a test observation will be assigned to the orange class, and the blue background grid indicates the region in which a test observation will be assigned to the blue class.*

Fonte: ISLR di Hastie & Tibshirani

Instance-based learning (es. K-nn)

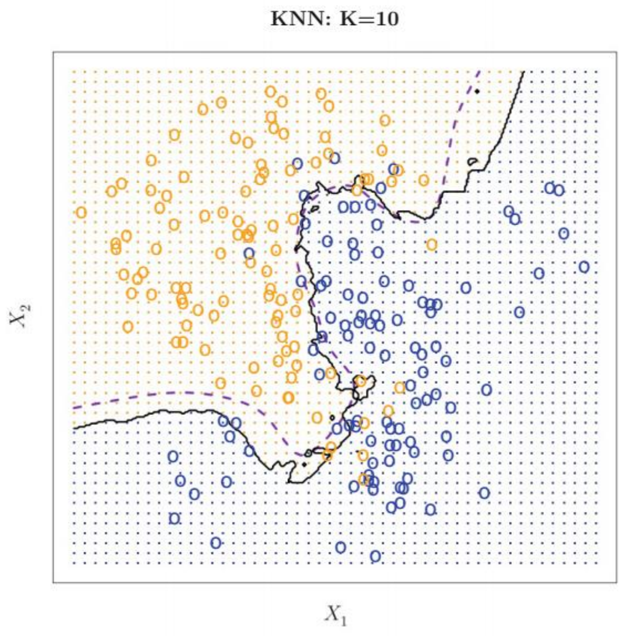


FIGURE 2.15. *The black curve indicates the KNN decision boundary on the data from Figure 2.13, using $K = 10$. The Bayes decision boundary is shown as a purple dashed line. The KNN and Bayes decision boundaries are very similar.*

La colorazione di queste griglie è fatta rispetto ai vari KNN, non rispetto a Bayes

overfitting **underfitting**

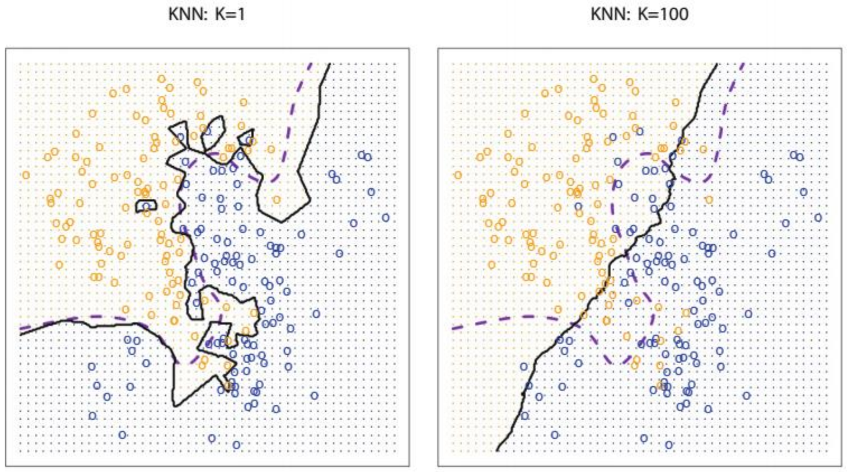


FIGURE 2.16. *A comparison of the KNN decision boundaries (solid black curves) obtained using $K = 1$ and $K = 100$ on the data from Figure 2.13. With $K = 1$, the decision boundary is overly flexible, while with $K = 100$ it is not sufficiently flexible. The Bayes decision boundary is shown as a purple dashed line.*

Fonte: ISLR di Hastie & Tibshirani

Esempio avanzato di uso di kNN: classificazione del tipo di terreno tramite immagine satellitari

(ESL di Hastie, Tibshiarni, Friedman, 2007)

- Un’area di territorio australiano <u>coltivato</u> composta da <u>sette</u> differenti tipi di terreno: rosso, misto, stoppie, grigio, grigio umido, grigio molto umido

- n immagini satellitari dell’area (nello spettro normale e nell’infrarosso)

- p è il numero di pixel di ogni immagine (82x100)

- Un problema di classificazione con 7 classi (per ogni pixel)

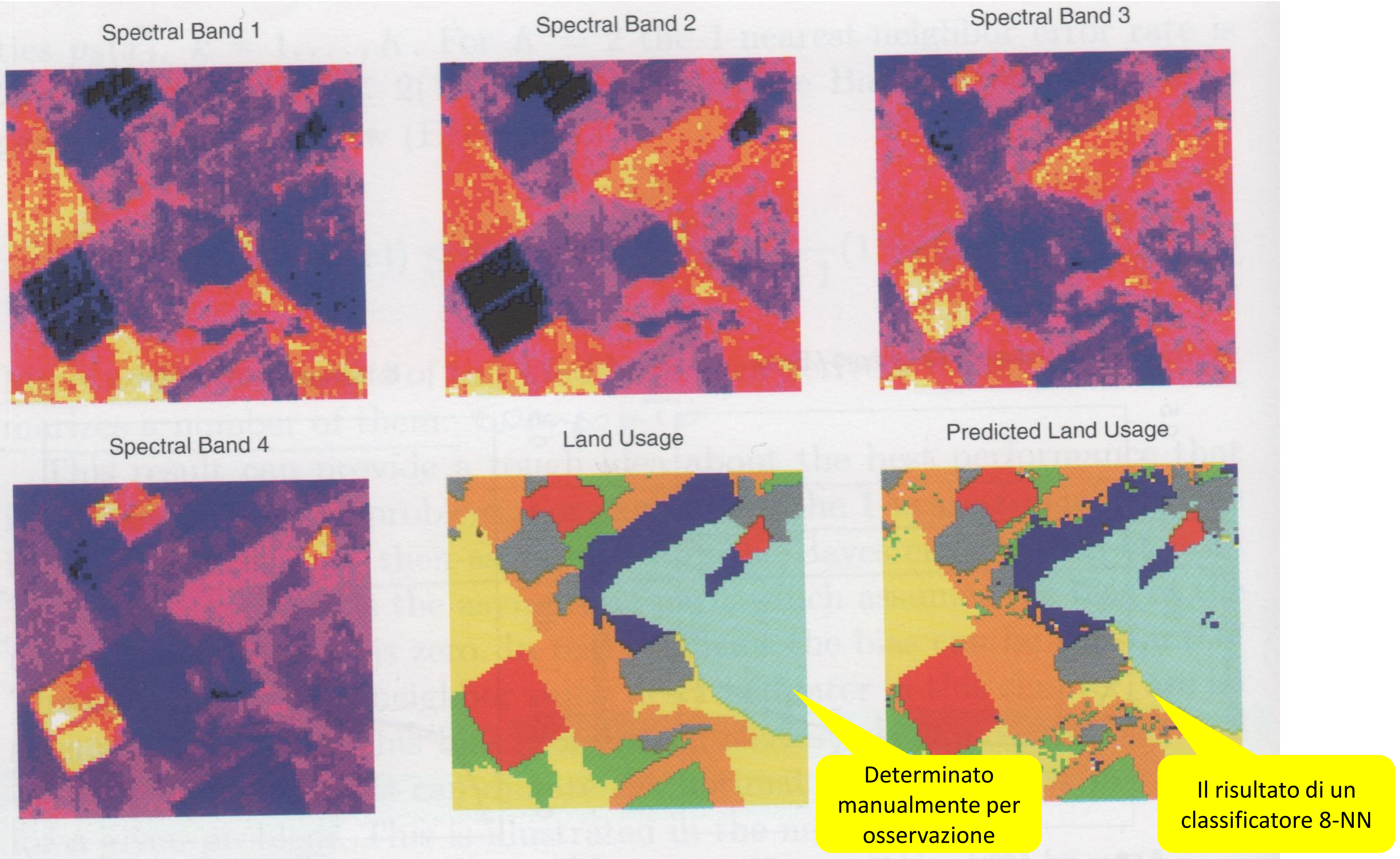


FIGURE 13.6. *The first four panels are LANDSAT images for an agricultural area in four spectral bands, depicted by heatmap shading. The remaining two panels give the actual land usage (color coded) and the predicted land usage using a five-nearest-neighbor rule described in the text.*

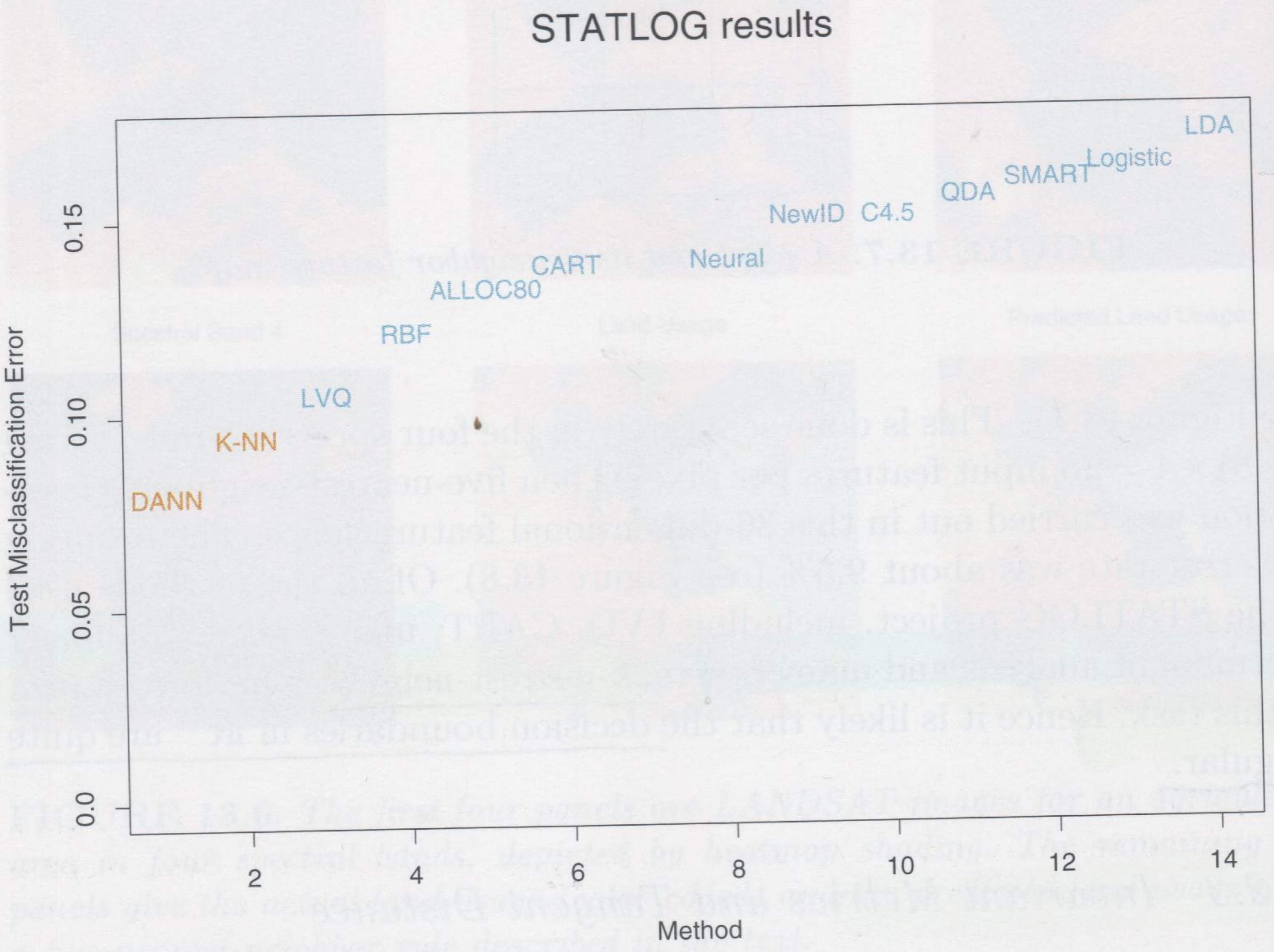
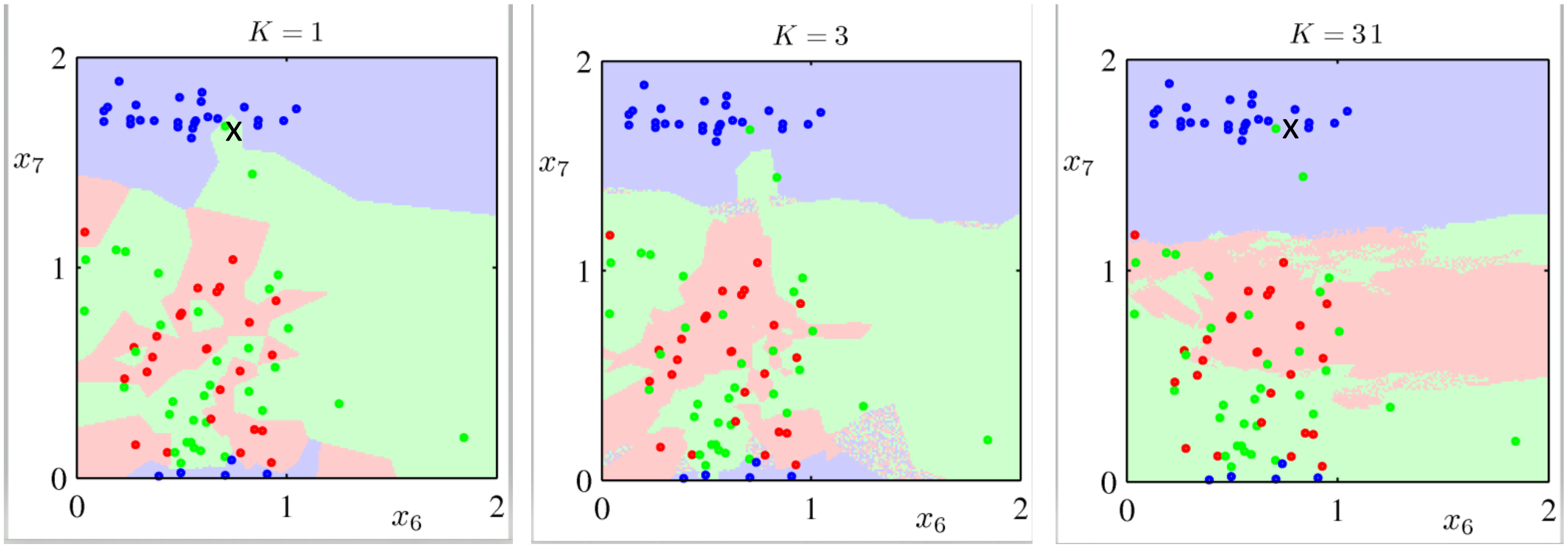


FIGURE 13.8. *Test-error performance for a number of classifiers, as reported by the STATLOG project. The entry DANN is a variant of k-nearest neighbors, using an adaptive metric (Section 13.4.2).*

I risultati dell’applicazione di KNN al problema dell’oleodotto



Plot di 200 punti l’uno

Il limite del kNN (ISLR, pp. 168-169)

When the number of features p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into the fact that non-parametric approaches often perform poorly when p is large. We will now investigate this curse.

(a) Suppose that we have a set of observations, each with measurements on p = 1 feature, X. We assume that X is uniformly (evenly) distributed on [0, 1]. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10% of the range of X closest to that test observation. For instance, in order to predict the response for a test observation with X = 0.6, we will use observations in the range [0.55, 0.65]. On average, what fraction of the available observations will we use to make the prediction?

(b) Now suppose that we have a set of observations, each with measurements on p = 2 features, X1 and X2. We assume that (X1, X2) are uniformly distributed on [0, 1] × [0, 1]. We wish to predict a test observation's response using only observations that are within 10% of the range of X1 and within 10% of the range of X2 closest to that test observation. For instance, in order to predict the response for a test observation with X1 = 0.6 and X2 = 0.35, we will use observations in the range [0.55, 0.65] for X1 and in the range [0.3, 0.4] for X2. On average, what fraction of the available observations will we use to make the prediction?

(c) Now suppose that we have a set of observations on p = 100 features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10% of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make the prediction?

(d) Using your answers to parts (a)–(c), argue that a drawback of KNN when p is large is that there are very few training observations "near" any given test observation.

(e) Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 10% of the training observations. For p = 1, 2, and 100, what is the length of each side of the hypercube? Comment on your answer.

Note: A hypercube is a generalization of a cube to an arbitrary number of dimensions. When p = 1, a hypercube is simply a line segment, when p = 2 it is a square, and when p = 100 it is a 100-dimensional cube.