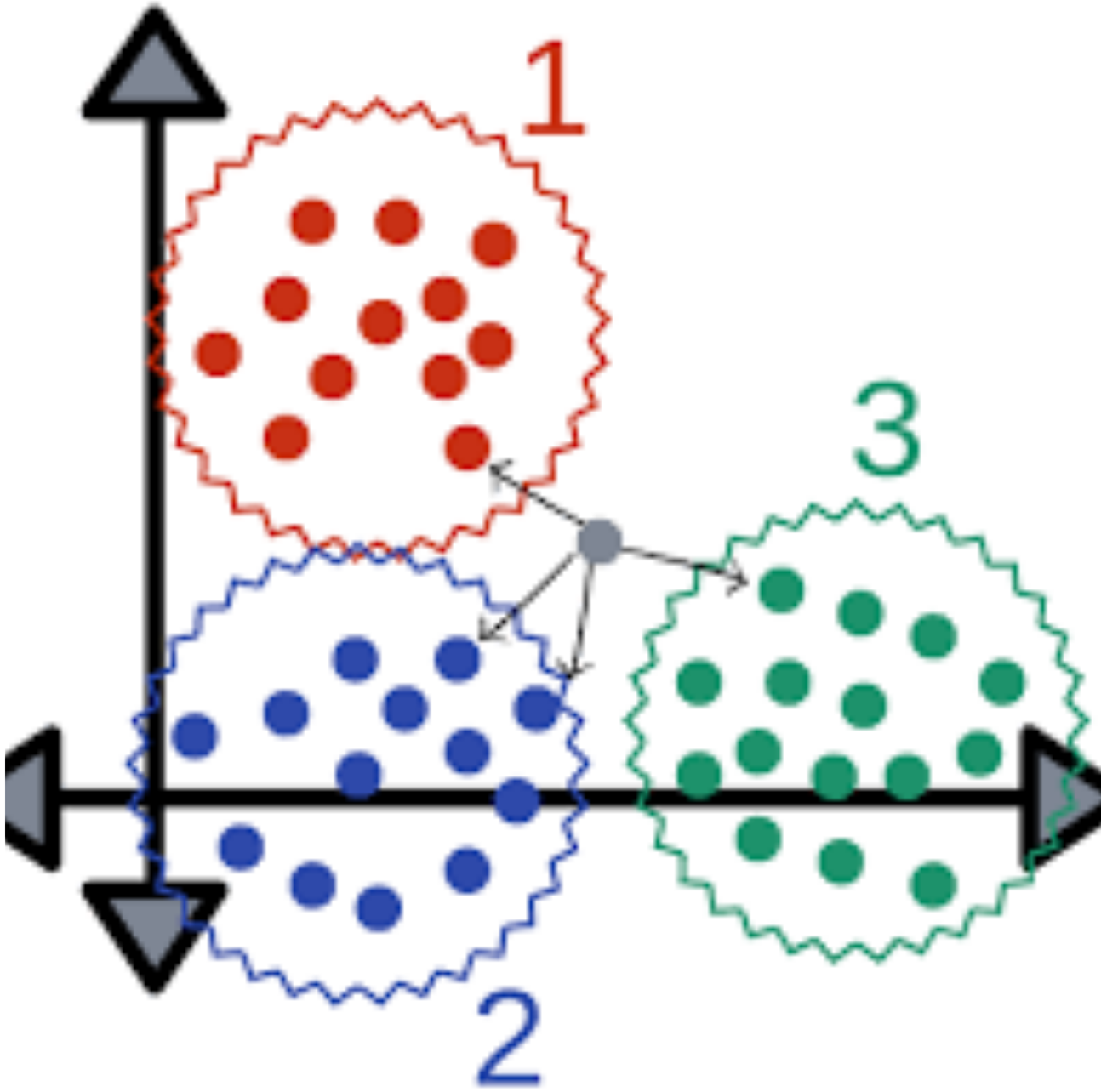
Data Science

Listo para convertirte en el mejor Data Science de la Historia?

Emanuel Lemos | Let’s coder

Clasificación

KNN / k-vecinos más cercanos



Clasificación

Introducción a la clasificación

- Problema de aprendizaje supervisado donde la variable objetivo es categórica
- Ejemplos de clasificación:clasificar emails en spam/no spam, identificar tipos de células cancerosas en malignas/benignas , etc.
- El modelo de clasificación es una función que mapea un set de atributos (X) a una clase (y)
- Los datos que usamos como input en una clasificación es una colección de instancias
- X es el set de atributos. Los atributos representan propiedades que pueden tomar valores continuos o discretos
- y es un atributo “especial” denominado clase (o atributo “target”).
- En una clasificación habitualmente la clase es categórica.

Clasificación

Introducción a la clasificación

Observaciones

Variable objetivo o etiqueta (y)

Features (X)

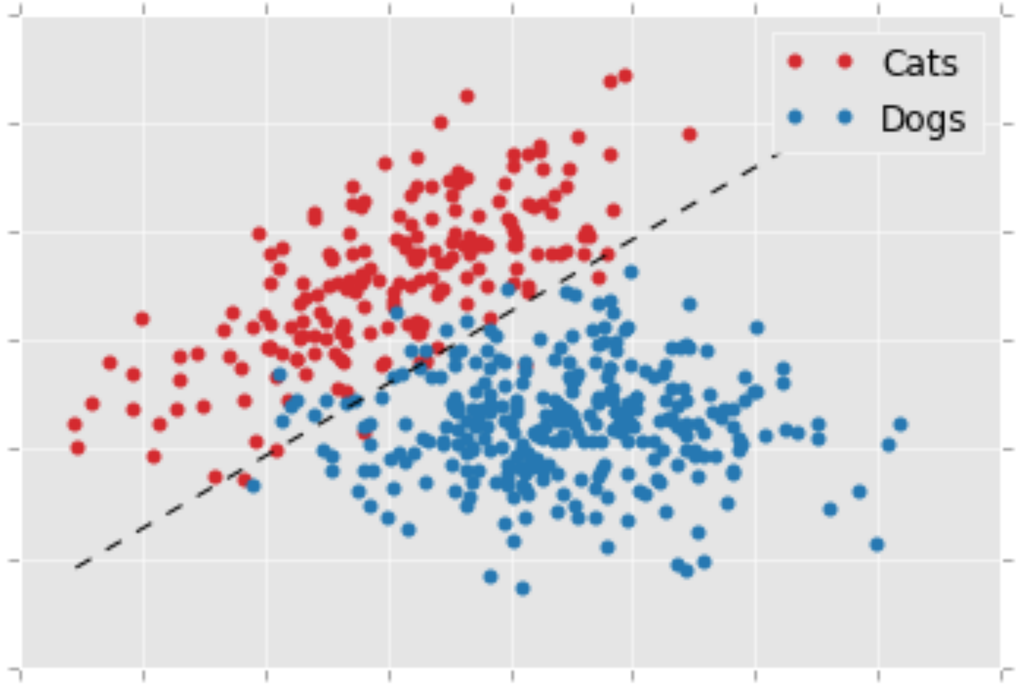
```
fruits.head(10)
```

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	178	7.1	7.8	0.92
9	1	apple	braeburn	172	7.4	7.0	0.89

```
fruits.shape
```

```
(59, 7)
```

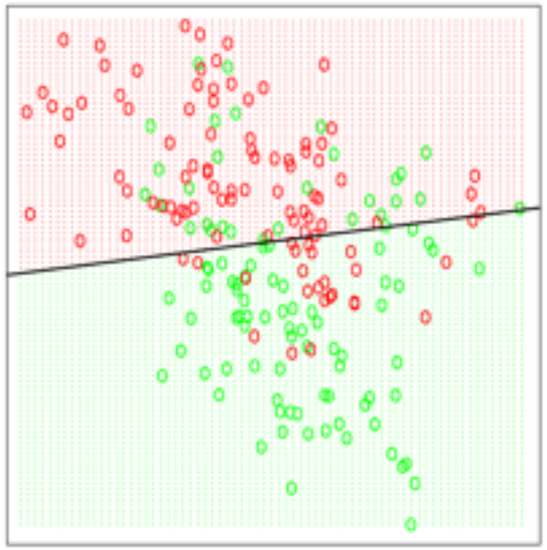
```
{1: 'apple', 2: 'mandarin', 3: 'orange', 4: 'lemon'}
```



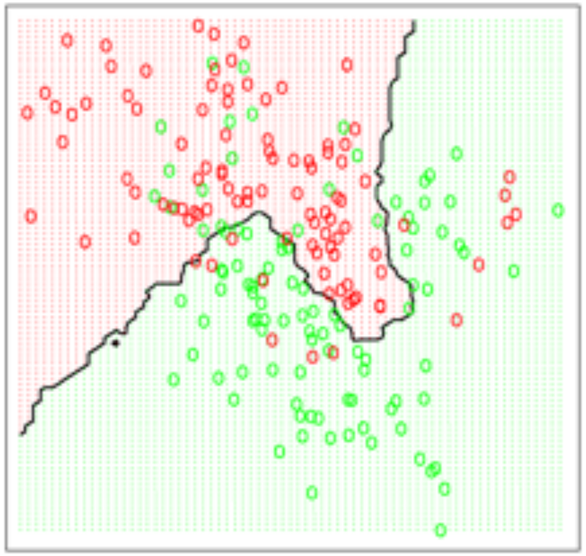
Clasificación

- Ejemplos de overfitting y underfitting

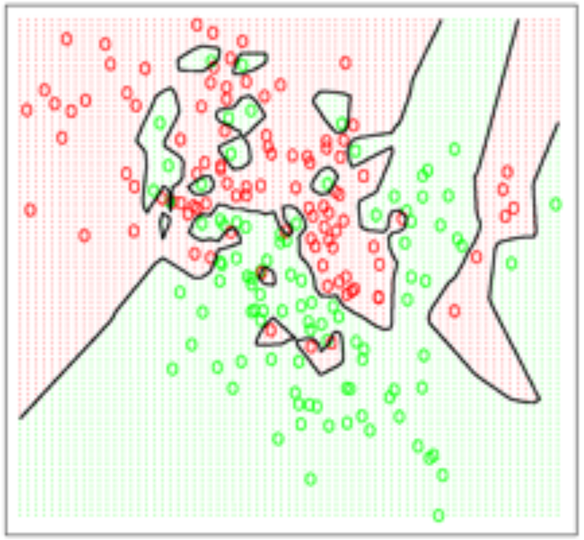
Underfit



Goodfit



Overfit



Clasificación

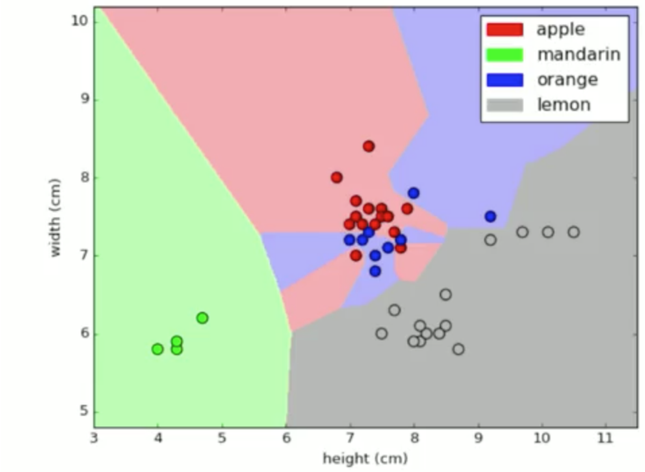
KNN

Clasificación
KNN

- Idea básica: un nuevo ejemplo se va a clasificar en la clase más frecuente a la que pertenecen sus K vecinos más cercanos, por la mayoría de los votos de sus vecinos
- La métrica de la vecindad de vecinos es una medida de similitud
  input = ejemplo / instancia no conocida
  Output = (label) membresía a un grupo predeterminado
- Pertenece al grupo de los métodos “non generalizing” o “instance-based” porque simplemente “recuerda” todos los datos de entrenamiento y con eso particiona el espacio para asignar la clasificación.

Clasificación
KNN

- Ante una nueva observación x_test para la cual hay que predecir la clase de pertenencia:
  -KNN busca en el set de entrenamiento a las k observaciones X_NN más cercanas a x_test.
  -Busca las etiquetas y_NN de X_NN
  -Predice la etiqueta de x_test combinando las etiquetas y_NN, por ejemplo usando voto de mayoría
- Ejemplo con k=1



Clasificación
KNN

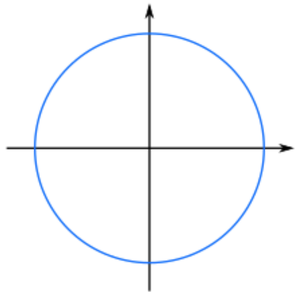
- Por default, se usa la **distancia euclidiana**, que funciona bien para la mayoría de los problemas. La distancia euclidiana es el caso especial de la métrica de Minkowsky con p=2.

The Minkowski distance of order *p* between two points

$$X = (x_1, x_2, \ldots, x_n) \text{ and } Y = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$$

is defined as:

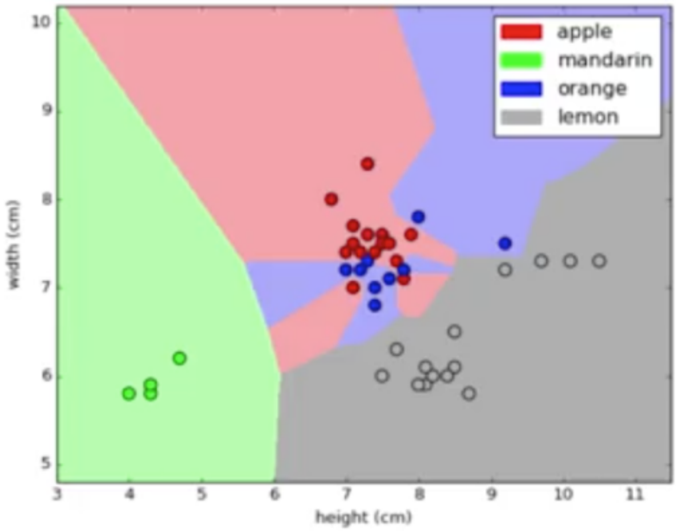
$$D(X, Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$



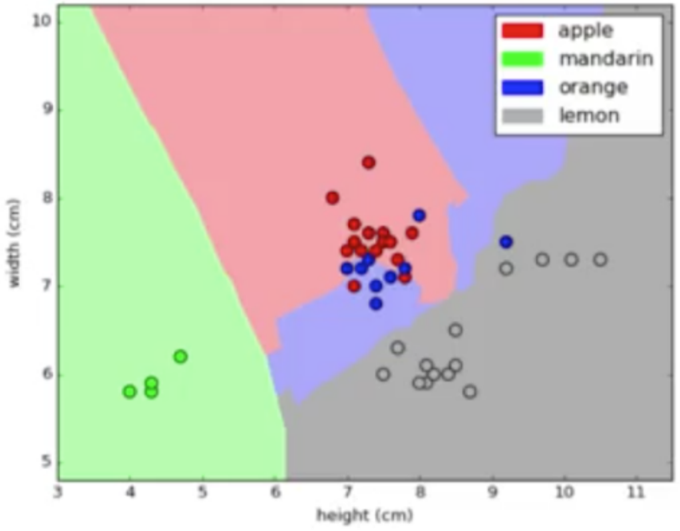
$$p = 2^1 = 2$$

Clasificación
KNN

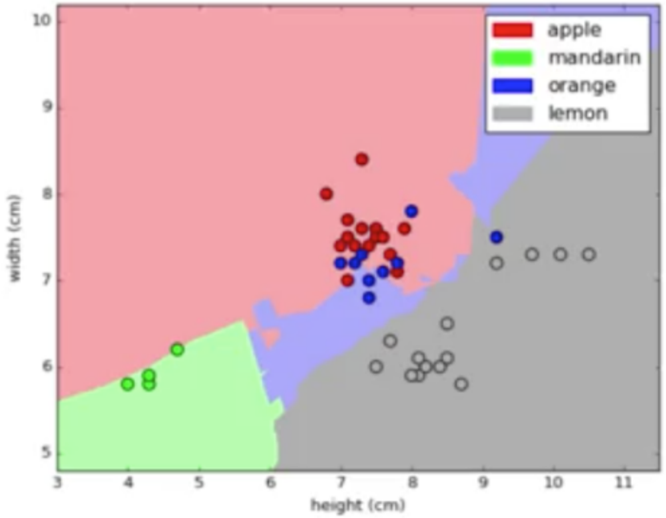
- El hiper parámetro K de este algoritmo es el que regula el trade-off entre sesgo y varianza



K=1



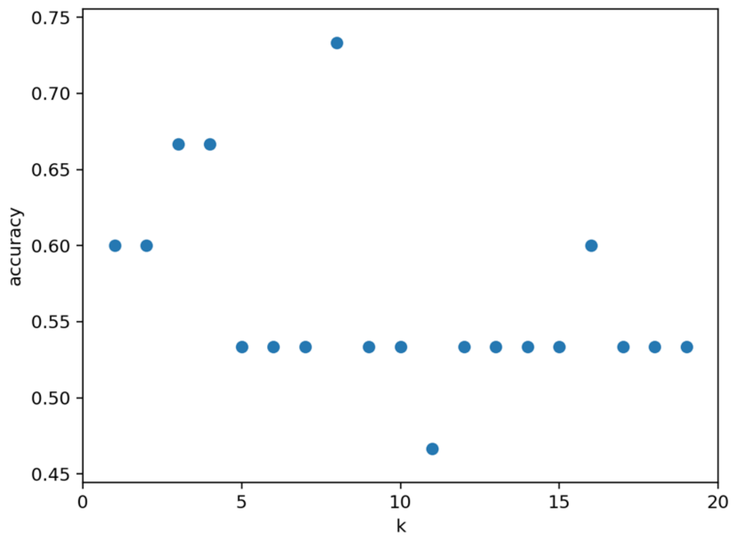
K=5



K=10

Clasificación
KNN

- Variación del accuracy en el test set al variar k




```
k_range = range(1,20)
scores = []

for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train, y_train)
    scores.append(knn.score(X_test, y_test))

plt.figure()
plt.xlabel('k')
plt.ylabel('accuracy')
plt.scatter(k_range, scores)
plt.xticks([0,5,10,15,20]);
```

Clasificación
KNN

- Conclusión:
- 1- Idea básica: un nuevo ejemplo se va a clasificar en la clase más frecuente a la que pertenecen sus **K vecinos más cercanos,** por la mayoría de los votos de sus vecinos
- 2- Pertenece al grupo de los métodos “non generalizing” o “**instance-based**” porque simplemente “recuerda” todos los datos de entrenamiento y con eso particiona el espacio para asignar la clasificación.
- 3- El **hiper parámetro K** de este algoritmo es el que regula el **trade-off entre sesgo y varianza**