Modelos Computacionales II

Clasificación de Vecinos más Cercanos

Mel, Anel, Andrea, Clau, Dafne, Annette, Rafa

Vecinos Cercanos

:)

-

Actividad

Vecinos Cercanos

¿Vecinos?
“Quiénes están más cerca” en cuanto a *similitudes*, entre lo que se esté comparando

¿Cercanos?
Esa parte la decide el sistema usando medidas parecidas a “cuántos pasos hay entre tú y la otra persona”- se refiere a *diferencias*

- Cuanto menos diferentes sean, más “cerca” están

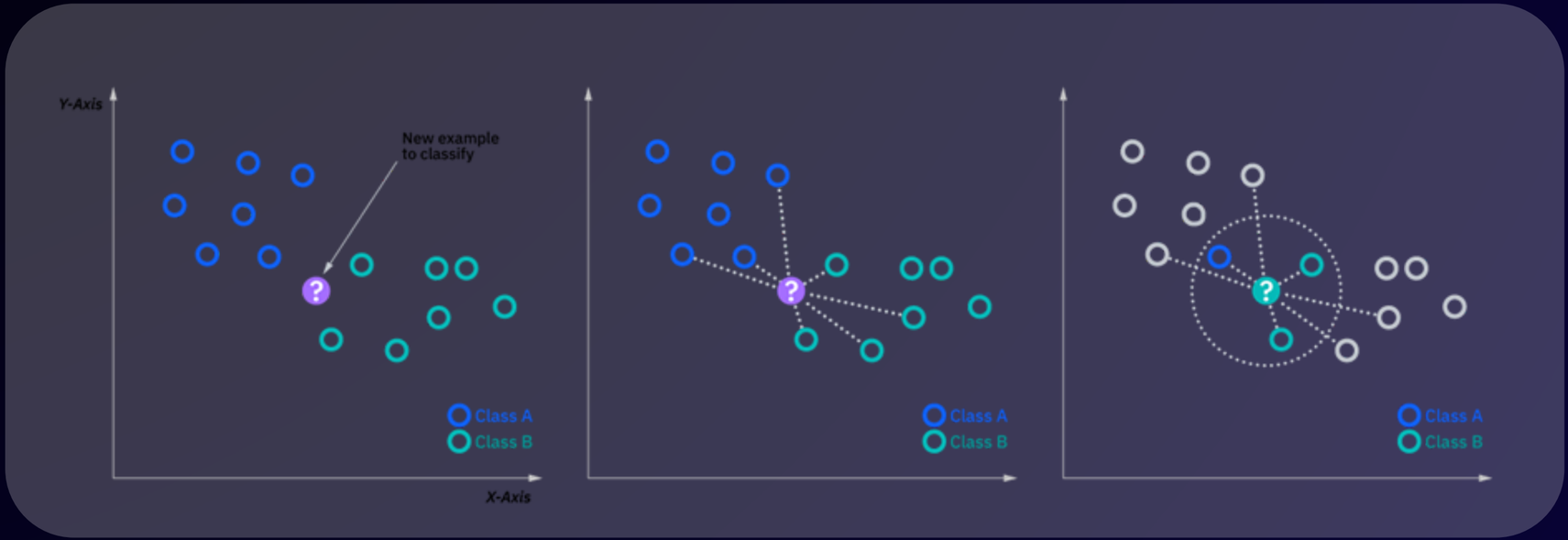
Ejemplos del mundo real

"Dime con quién andas y te diré quién eres"

¿Qué es?

Clasificador de aprendizaje supervisado que funciona basándose en la similitud entre los datos.

Para clasificar o predecir un dato nuevo, KNN compara esa instancia con sus "k" vecinos más cercanos en el conjunto de entrenamiento y realiza la predicción en función de las categorías (clasificación) o valores (regresión) de esos vecinos.



Definir K

K es el número de vecinos más cercanos que el algoritmo va a considerar para tomar una decisión.

Un valor adecuado de k balancea el sobreajuste y el subajuste.

- **Valor pequeño de k:**
  - Muy sensible al ruido y a los valores atípicos.
  - **Riesgo de sobreajuste:** El modelo se ajusta demasiado a los datos de entrenamiento.

- **Valor grande de k:**
  - Pierde detalles importantes de los datos locales.
  - **Riesgo de subajuste:** El modelo generaliza demasiado y pierde precisión.

Clasificación y Regresión

Clasificación:

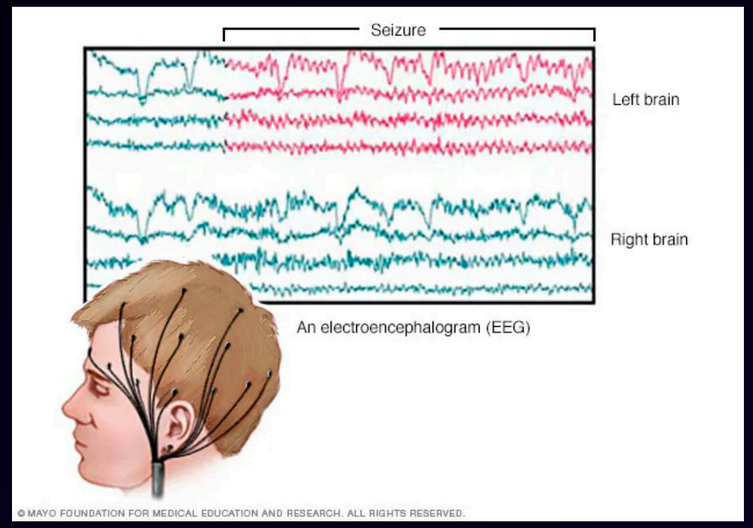
- Predecir una categoría
- El resultado es discreto (Ejemplo: spam / no spam, gato / perro).
- KNN asigna la clase más común entre los vecinos cercanos (votación de mayoría).

Regresión:

- Predice valores numéricos continuos.
- El resultado es un número (por ejemplo: precio, temperatura, tiempo).
- KNN calcula el promedio de los valores de los vecinos más cercanos.

APLICACIONES DEL KNN EN LAS NEUROCIENCIAS

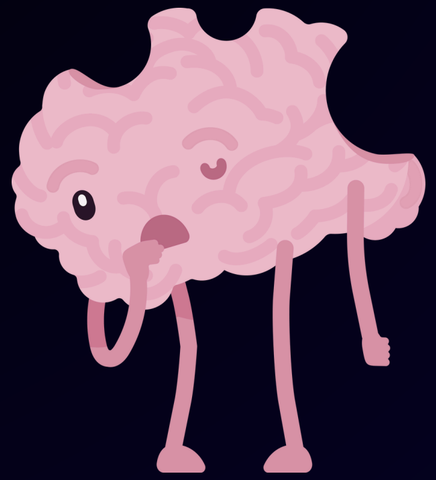
- # Clasificación de estados mentales con EEG



se extraen características como la potencia en bandas (alfa, beta) y se asocian con un estado conocido (sueño, vigilia, anestesiado)
KNN compara las nuevas muestras con las etiquetadas previamente

KNEIGHBORSCLASSIFIER

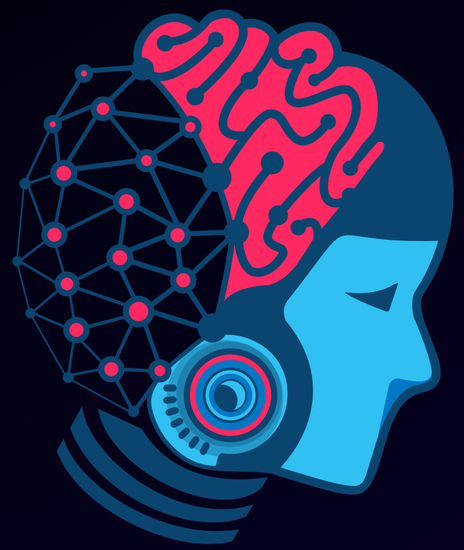
- **Diagnóstico de trastornos neurologicos**



GRIDSEARCHCV

A partir de datos clínicos (edad, volumen cerebral, score cognitivo) se crean etiquetas como paciente sano, Párkinson o Alzheimer y se entrena al modelo para que clasifique a que grupo pertenece un nuevo sujeto.

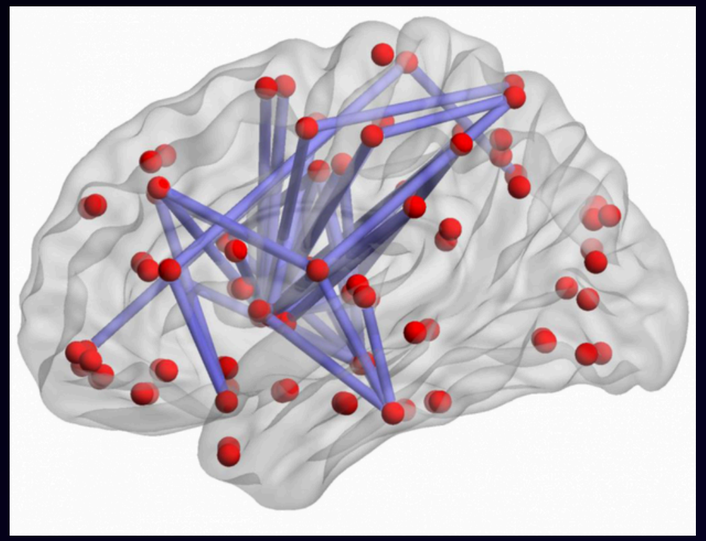
- # Interfaz cerebro-computadora



Durante el entrenamiento se asocian señales cerebrales a intenciones motoras (pensar en mover la mano derecha) y se extraen características la activacion de frecuencias o patrones especiales

BUFFER DE VENTA MÓVIL (TIEMPO REAL)

- **Conectividad cerebral**



A partir de matrices de correlación, se obtienen características del grado como numero de conexiones o eficiencia, y cada individuo diagnosticado genera su vector de características y se compara con las nuevas redes cerebrales

ÚTIL CON POCOS SUJETOS Y ALTA DIMENSIONALIDAD

VENTAJAS VS DESVENTAJAS

- Simple
- “algoritmo perezoso”
- Adaptable a problemas complejos
- Versatilidad
- Rendimiento competitivo en datos bien distribuidos

- Lento en datasets demasiado grandes
- sensibilidad al ruido y a los datos irrelevantes
- gran cosumo de memoria

Código*

Clasificación de flores con Iris dataset.

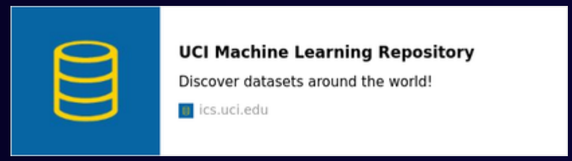
Se busca predecir la especie de flor (setosa, versicolor, virginica) según el ancho y largo del pétalo y sépalo. Usar K-Nearest Neighbors (KNN) que predice la especie de una flor basada en sus características, y ver cómo cambia la exactitud al cambiar el número de vecinos

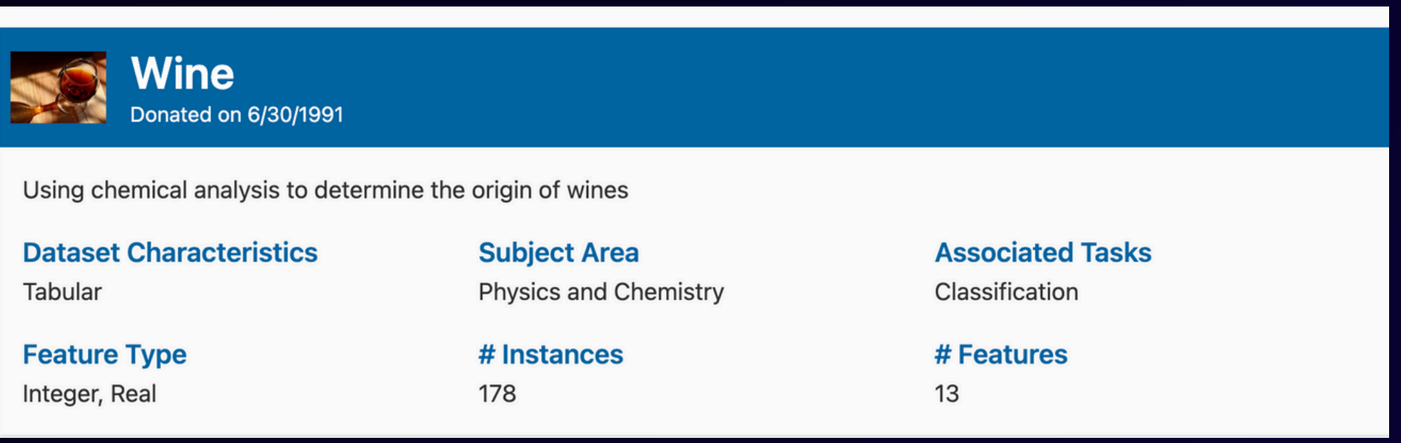
***Jupyter Notebook !**

```python
import pandas as pd
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score

# Load Wine dataset
wine = load_wine()
wine_df = pd.DataFrame(data=wine.data, columns=wine.feature_names)
wine_df['target'] = wine.target

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
    wine.data, wine.target, test_size=0.3, random_state=42
)
```





```
# KNN classifier with 3 neighbors
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

# Predict the test set
y_pred = knn.predict(X_test)

# Evaluate model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print(classification_report(y_test, y_pred, target_names=wine.target_names
```

	precision	recall	f1-score	support
class_0	0.89	0.89	0.89	19
class_1	0.75	0.71	0.73	21
class_2	0.53	0.57	0.55	14
accuracy			0.74	54
macro avg	0.73	0.73	0.73	54
weighted avg	0.74	0.74	0.74	54

Bibliografía

- https://www.elastic.co/es/what-is/knn
- https://www.ibm.com/mx-es/think/topics/knn

Thank You