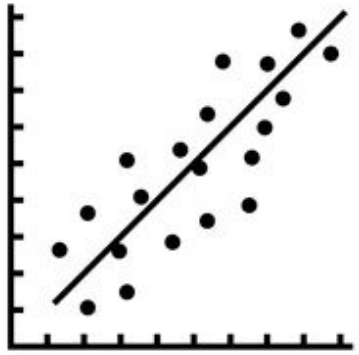
Clasificación de sistemas de aprendizaje automático

Práctica

Profesor:
David Campoy Miñarro

Aprendizaje automático supervisado

Regresión



Aprendizaje automático supervisado

El algoritmo puede aprender y obtener una función que sea capaz de generalizar y predecir la respuesta para un conjunto de valores nuevo.

Regresión

Clasificación

Tenemos el siguiente dataset:

Disponemos de un Dataset sobre viviendas de Boston. Imagina que pudiéramos tener uno sobre Benidorm.

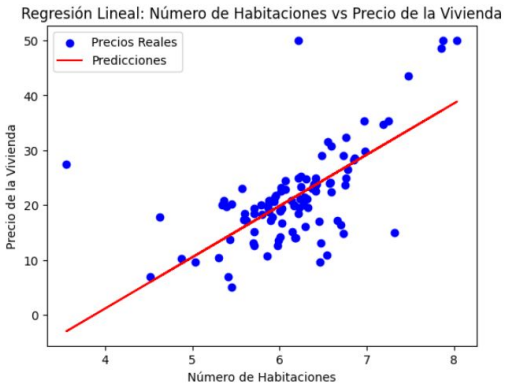
Estos atributos ofrecen una diversidad de información socioeconómica y geográfica que se utilizan comúnmente para la predicción de precios de viviendas en modelos de regresión.

Por ejemplo, la cantidad de habitaciones (RM), la tasa de criminalidad (CRIM), la proximidad a centros de empleo (DIS) y el porcentaje de estatus bajo de la población (LSTAT) son atributos que pueden desempeñar un papel significativo en la determinación de los precios de las viviendas.

- **CRIM:** Tasa de criminalidad per cápita por área.
- **ZN:** Proporción de terreno residencial dividido en zonas para lotes mayores a 25,000 pies cuadrados.
- **INDUS:** Proporción de acres de negocios no minoristas por ciudad.
- **CHAS:** Variable ficticia de Charles River (= 1 si el tramo limita con el río; 0 en caso contrario).
- **NOX:** Concentración de óxidos nítricos (partes por 10 millones).
- **RM:** Número medio de habitaciones por vivienda.
- **AGE:** Proporción de unidades ocupadas por sus propietarios construidas antes de 1940.
- **DIS:** Distancias ponderadas a cinco centros de empleo de Boston.
- **RAD:** Índice de accesibilidad a carreteras radiales.
- **TAX:** Tasa de impuesto a la propiedad por $10,000.
- **PTRATIO:** Proporción alumno-maestro por localidad.
- **B:** Proporción de personas de ascendencia afroamericana por ciudad.
- **LSTAT:** Porcentaje de población de estatus bajo.
- **MEDV:** Valor medio de las viviendas ocupadas por sus propietarios en miles de dólares.

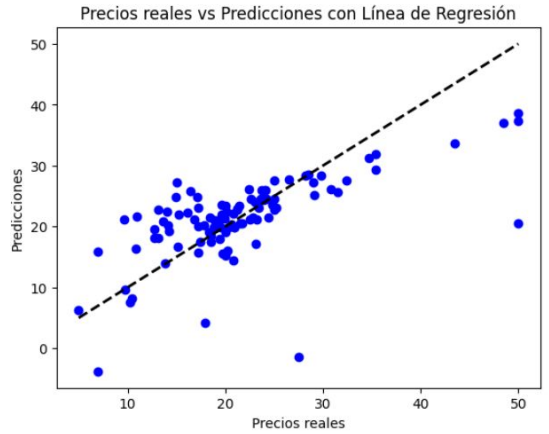
```python
from sklearn.model_selection import train_test_split  # Importar función para dividir datos en conjuntos de entrenamiento y prueba
from sklearn.linear_model import LinearRegression  # Importar el modelo de regresión lineal
import matplotlib.pyplot as plt  # Importar la librería para visualización
import pandas as pd  # Importar pandas para manejo de datos

# Cargar el conjunto de datos de viviendas de Boston utilizando pandas
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data"
names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
boston = pd.read_csv(url, delim_whitespace=True, names=names)

X = boston.drop('MEDV', axis=1)  # Características de las viviendas
y = boston['MEDV']  # Precios de las viviendas

# Elegir una única característica para visualización (por ejemplo, número de habitaciones)
X_rooms = X[['RM']]  # Usar únicamente el número de habitaciones

# Dividir los datos en conjuntos de entrenamiento y prueba (80% entrenamiento, 20% prueba)
X_train, X_test, y_train, y_test = train_test_split(X_rooms, y, test_size=0.2, random_state=42)

# Crear un modelo de regresión lineal
modelo = LinearRegression()

# Entrenar el modelo usando los conjuntos de entrenamiento
modelo.fit(X_train, y_train)

# Realizar predicciones usando el conjunto de prueba
predicciones = modelo.predict(X_test)

# Visualizar los resultados (comparando precios reales vs predicciones)
plt.scatter(X_test, y_test, color='blue', label='Precios Reales')
plt.plot(X_test, predicciones, color='red', label='Predicciones')
plt.xlabel('Número de Habitaciones')
plt.ylabel('Precio de la Vivienda')
plt.title('Regresión Lineal: Número de Habitaciones vs Precio de la Vivienda')
plt.legend()
plt.show()
```

Este ejemplo predice el valor de una vivienda en función del número de habitaciones.



```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import pandas as pd

# Cargar el conjunto de datos de viviendas de Boston usando pandas
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data"
names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
boston = pd.read_csv(url, delim_whitespace=True, names=names)

# Seleccionar las características y el precio de las viviendas
X = boston[['CRIM', 'RM']]  # Características: Tasa de criminalidad y Número medio de habitaciones
y = boston['MEDV']  # Precio de las viviendas

# Dividir los datos en conjuntos de entrenamiento y prueba (80% entrenamiento, 20% prueba)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Crear un modelo de regresión lineal
modelo = LinearRegression()

# Entrenar el modelo usando los conjuntos de entrenamiento
modelo.fit(X_train, y_train)

# Realizar predicciones usando el conjunto de prueba
predicciones = modelo.predict(X_test)

# Visualizar los resultados (comparando precios reales vs predicciones con línea de regresión)
plt.scatter(y_test, predicciones, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)  # Línea de regresión ideal
plt.xlabel('Precios reales')
plt.ylabel('Predicciones')
plt.title('Precios reales vs Predicciones con Línea de Regresión')
plt.show()
```

Si los puntos se encuentran cercanos a la línea de pendiente 45 grados, significa que las predicciones del modelo son bastante cercanas a los valores reales. En cambio, si los puntos se dispersan ampliamente lejos de esta línea, indica que las predicciones difieren significativamente de los valores reales.



```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import r2_score

# Cargar el conjunto de datos de viviendas de Boston usando pandas
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data"
names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
boston = pd.read_csv(url, delim_whitespace=True, names=names)

# Seleccionar las características y el precio de las viviendas
X = boston[['CRIM', 'RM']]
y = boston['MEDV']  # Precio de las viviendas

# Dividir los datos en conjuntos de entrenamiento y prueba (80% entrenamiento, 20% prueba)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Crear un modelo de regresión lineal
modelo = LinearRegression()

# Entrenar el modelo usando los conjuntos de entrenamiento
modelo.fit(X_train, y_train)

# Realizar predicciones usando el conjunto de prueba
predicciones = modelo.predict(X_test)

# Calcular el coeficiente de determinación (R²)
r2 = r2_score(y_test, predicciones)
print(f"El coeficiente de determinación, R² es: {r2}")

# Visualizar los resultados (comparando precios reales vs predicciones con línea de regresión)
plt.scatter(y_test, predicciones, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)  # Línea de regresión ideal
plt.xlabel('Precios reales')
plt.ylabel('Predicciones')
plt.title('Precios reales vs Predicciones con Línea de Regresión')
plt.show()
```

Este código calculará el **coeficiente de determinación (R^2)** para evaluar el modelo de regresión lineal.

Cuanto más cercano a 1 sea el valor de R^2, mejor será la capacidad del modelo para predecir los precios de las viviendas.

Actividad:

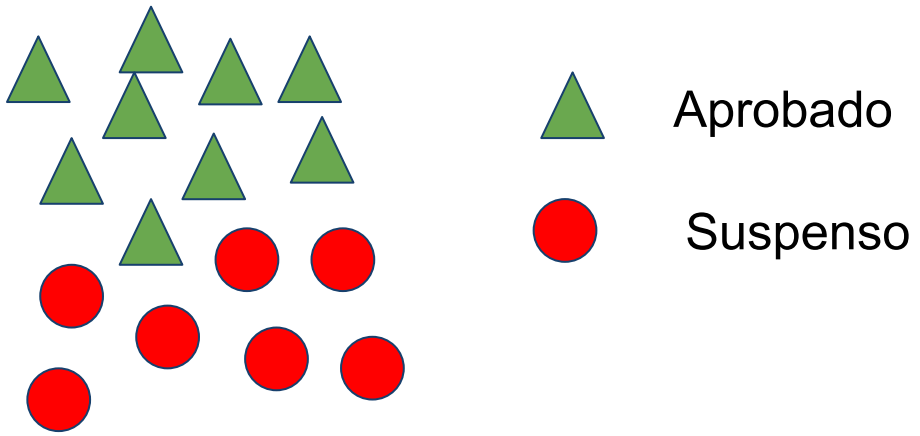
Prueba a combinar características de las viviendas y comprueba cómo varía el coeficiente de determinación.

Busca los parámetros con el coeficiente de determinación más alto.

Ese será tu objetivo como experto en IA.

Aprendizaje automático supervisado

Clasificación



Aprendizaje automático supervisado

El algoritmo puede aprender y obtener una función que sea capaz de generalizar y predecir la respuesta para un conjunto de valores nuevo.

Regresión

Clasificación

Tenemos el siguiente dataset del Titanic:

Con este dataset vamos a clasificar los pasajeros supervivientes de los no supervivientes.

Es decir, ¿quieres saber que posibilidades tienes de sobrevivir a un naufrágio? ¿Dónde estarías, en superviviente o no superviviente?

Pues un sistema de aprendizaje automático va a intentar saberlo.

Ahora, imagina utilizar datos de tu empresa.

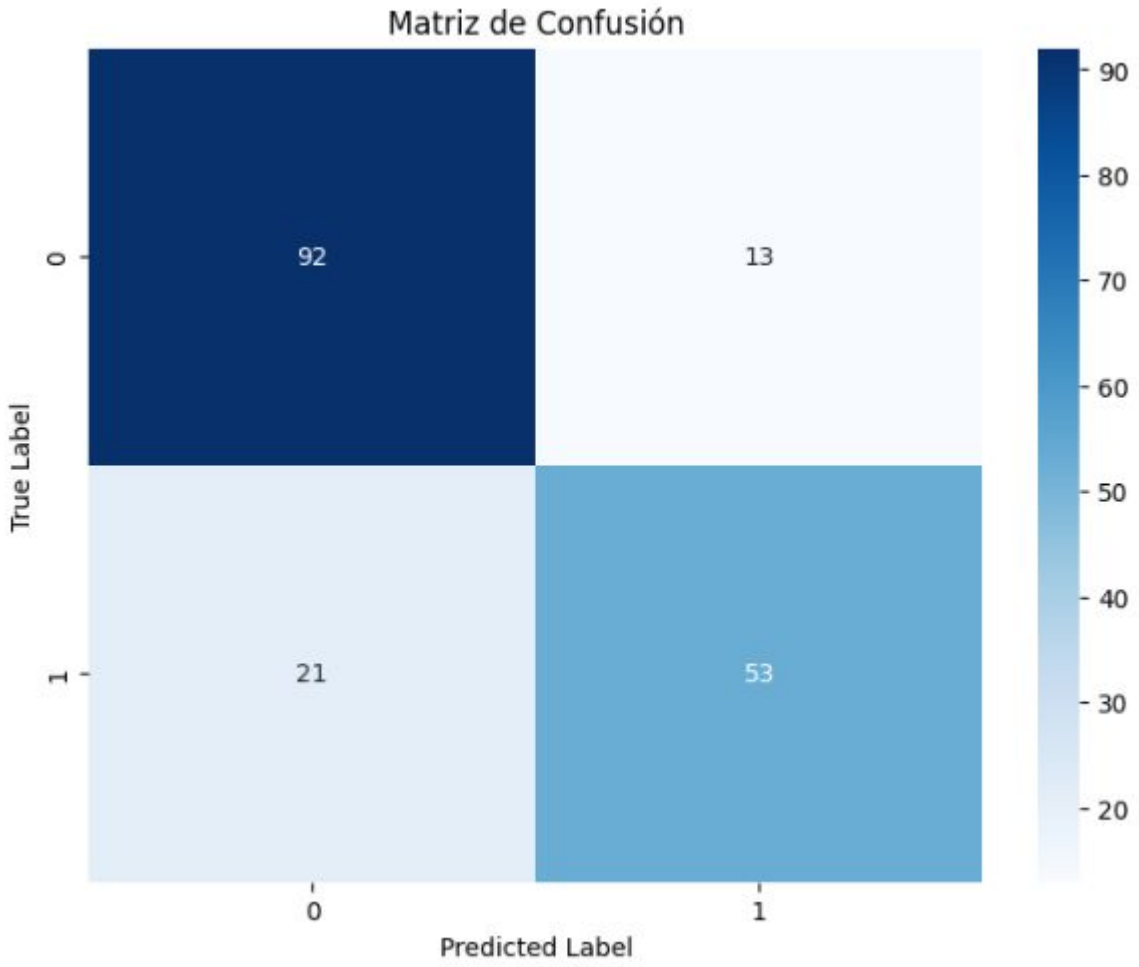
La Regresión Logística es un algoritmo de aprendizaje automático comúnmente utilizado para problemas de clasificación binaria.

- **Pclass (Clase socioeconómica)**: Representa la clase en la que viajaba el pasajero (1 = Primera clase, 2 = Segunda clase, 3 = Tercera clase).
- **Age (Edad)**: La edad del pasajero.
- **SibSp:** Número de hermanos/cónyuges a bordo del Titanic.
- **Parch:** Número de padres/hijos a bordo del Titanic.
- **Fare (Tarifa):** El precio del ticket.
- **Sex_male (Género masculino):** Una variable que indica si el pasajero es masculino (1) o no (0).

`https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv`

```
# Importar las librerías necesarias
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt

# Cargar el conjunto de datos del Titanic
url =
"https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv
"
data = pd.read_csv(url)

# Visualizar las primeras filas del dataset
print(data.head())

# Eliminar columnas irrelevantes o con datos faltantes
data = data.drop(['Name', 'Ticket', 'Cabin', 'Embarked'], axis=1)

# Manejar datos faltantes (en este caso, rellenar los valores faltantes con la
media)
imputer = SimpleImputer(strategy='mean')
data['Age'] = imputer.fit_transform(data['Age'].values.reshape(-1, 1))

# Convertir variables categóricas a variables dummy
data = pd.get_dummies(data, columns=['Sex'], drop_first=True)

# Dividir los datos en características (features) y etiquetas (labels)
X = data.drop('Survived', axis=1)
y = data['Survived']

# Dividir los datos en conjunto de entrenamiento y conjunto de prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Crear el modelo de clasificación (Regresión Logística)
model = LogisticRegression(max_iter=1000)

# Entrenar el modelo
model.fit(X_train, y_train)

# Realizar predicciones
predictions = model.predict(X_test)

# Evaluar el modelo
accuracy = accuracy_score(y_test, predictions)
report = classification_report(y_test, predictions)

# Visualización: Histograma de edades
plt.figure(figsize=(8, 6))
plt.hist(data[data['Survived'] == 1]['Age'], bins=30, alpha=0.5,
label='Sobrevivientes')
plt.hist(data[data['Survived'] == 0]['Age'], bins=30, alpha=0.5,
label='No sobrevivientes')
plt.xlabel('Edad')
plt.ylabel('Cantidad de Pasajeros')
plt.title('Distribución de Edades de Sobrevivientes y No
Sobrevivientes')
plt.legend()
plt.show()

# Visualización: Gráfico de barras de la cantidad de sobrevivientes
por género
survived_gender = data.groupby('Sex_male')['Survived'].sum()
plt.figure(figsize=(6, 6))
survived_gender.plot(kind='bar', color=['skyblue', 'pink'])
plt.xticks([0, 1], ['Mujer', 'Hombre'], rotation=0)
plt.xlabel('Género')
plt.ylabel('Cantidad de Sobrevivientes')
plt.title('Cantidad de Sobrevivientes por Género')
plt.show()

# Imprimir la precisión del modelo y el informe de clasificación
print(f'Precisión del modelo: {accuracy}')
print('Informe de clasificación:')
print(report)
```

Resultados




```
Precisión del modelo: 0.8100558659217877
Informe de clasificación:
              precision    recall  f1-score   support

           0       0.81      0.88      0.84       105
           1       0.80      0.72      0.76        74

    accuracy                           0.81       179
   macro avg       0.81      0.80      0.80       179
weighted avg       0.81      0.81      0.81       179

Matriz de Confusión:
[[92 13]
 [21 53]]
```

Actividad:

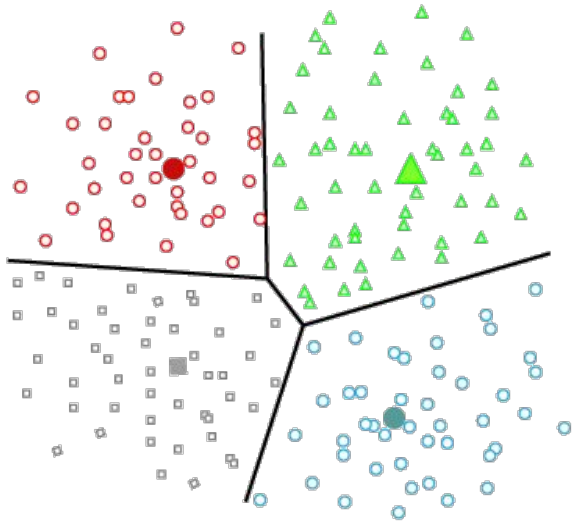
- Interpreta la matriz de confusión.
- ¿Cómo influye la edad en la categorización?
- ¿Y el precio del billete?
- Documenta todas las pruebas que has realizado.

También puede cambiar los párametros:

- test_size=0.2
- max_iter=1000

Aprendizaje automático no supervisado

Clustering



Profesor: David Campoy Miñarro

Clustering

Partimos de un dataset sin etiquetar, siendo su objetivo el de agrupamiento, por lo que el algoritmo hace grupos de elementos similares por sí solo, sin ningún tipo de conocimiento. Este algoritmo identifica patrones que le permiten separar en grupos.

El dataset que vamos a utilizar es el conjunto de datos "Mall Customers" que contiene información sobre clientes de un centro comercial.

Este conjunto de datos incluye cinco columnas:

- **CustomerID**: Identificador único para cada cliente.
- **Gender**: Género del cliente (Masculino o Femenino).
- **Age**: Edad del cliente.
- **Annual Income**: Ingreso anual del cliente.
- **Spending Score**: Puntaje asignado según el comportamiento del cliente en el centro comercial.

Realizar el clustering de clientes basado en dos características: el ingreso anual y el puntaje de gasto.

Con el algoritmo de K-Means, intentarems agrupar a los clientes en diferentes segmentos o clusters de manera que los clientes dentro de cada cluster sean similares en **términos de ingresos** y **puntajes de gasto**.

Esto puede ayudar a comprender mejor el comportamiento de los clientes y a tomar decisiones comerciales estratégicas, como la **segmentación de mercado** o la **personalización de servicios**.

`https://gist.github.com/pravalliyaram/5c05f43d2351249927b8a3f3cc3e5ecf`

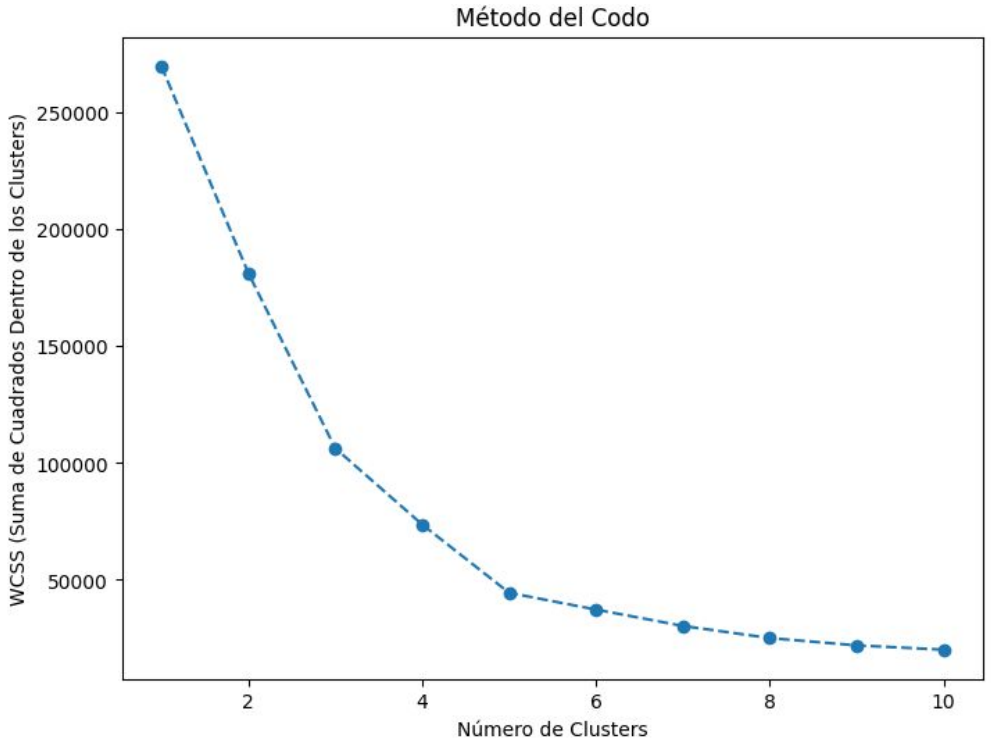
```
# Importar las bibliotecas necesarias
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Cargar el conjunto de datos "Mall Customers"
data = pd.read_csv('./Mall_Customers.csv')

# Mostrar las primeras filas del conjunto de datos
print(data.head())

# Seleccionar las características relevantes para el clustering
X = data.iloc[:, [3, 4]].values  # Estamos usando 'Annual
Income' y 'Spending Score'

# Encontrar el número óptimo de clusters usando el método del
codo (Elbow Method)
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', n_init=10,
random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Graficar el método del codo para determinar el número óptimo
de clusters
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.title('Método del Codo')
plt.xlabel('Número de Clusters')
plt.ylabel('WCSS (Suma de Cuadrados Dentro de los Clusters)')
plt.show()
```

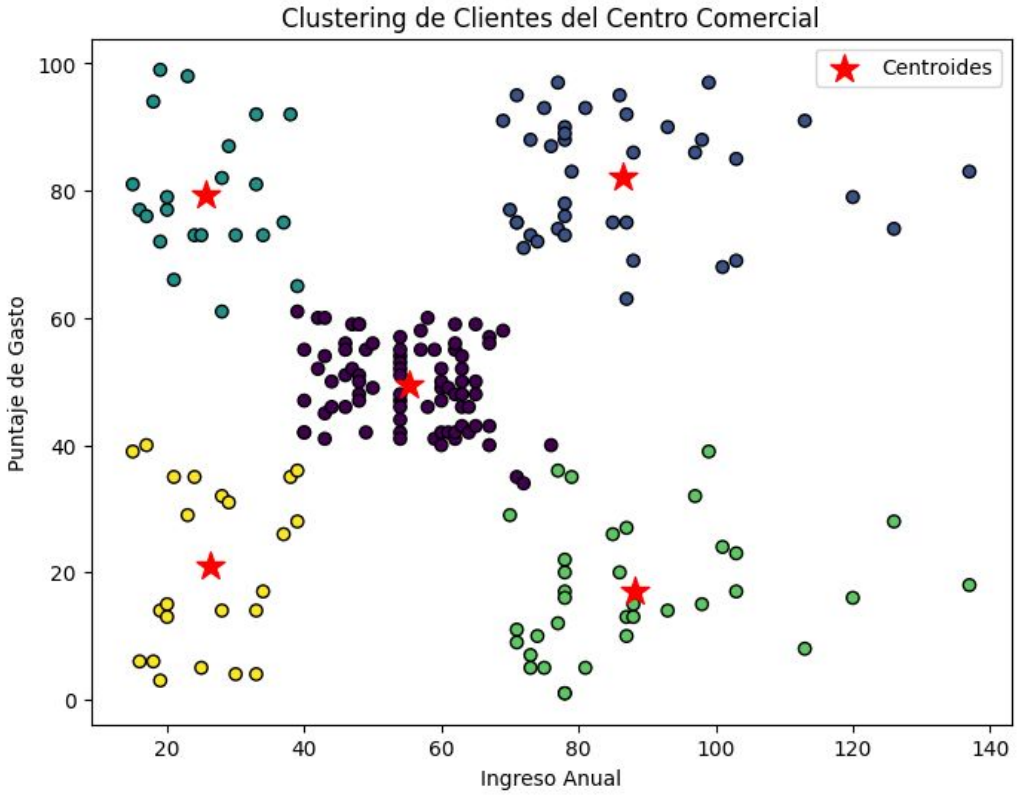
```
# Aplicar K-Means con el número óptimo de clusters
(determinado visualmente)
kmeans = KMeans(n_clusters=5, init='k-means++',
n_init=10, random_state=42)
kmeans.fit(X)

# Visualizar los clusters
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_,
cmap='viridis', edgecolor='k')
plt.scatter(kmeans.cluster_centers_[:, 0],
kmeans.cluster_centers_[:, 1], marker='*', s=200,
c='red', label='Centroides')
plt.title('Clustering de Clientes del Centro Comercial')
plt.xlabel('Ingreso Anual')
plt.ylabel('Puntaje de Gasto')
plt.legend()
plt.show()
```

K-Means es un algoritmo de aprendizaje automático no supervisado utilizado para realizar tareas de clustering.

El objetivo principal del algoritmo K-Means es dividir un conjunto de datos en K clusters, donde K es un número predefinido por el usuario.

Resultados





Método del Codo
Esta técnica busca determinar el número óptimo de clusters (K) para el algoritmo K-Means. Muestra la suma de los cuadrados de las distancias entre los puntos y los centroides del cluster a medida que aumentamos el número de clusters. Se busca el punto en el gráfico donde la tasa de disminución de la suma de cuadrados ya no es significativa, formando un codo. En este caso, se observa el codo en 5 clusters.

Gráfico clustering
Cada punto en el gráfico representa a un cliente y está ubicado en un espacio bidimensional definido por su ingreso anual y su puntaje de gasto. Los puntos están coloreados según el cluster al que pertenecen. Los centroides de cada cluster se representan con estrellas rojas.

Aprendizaje semisupervisado

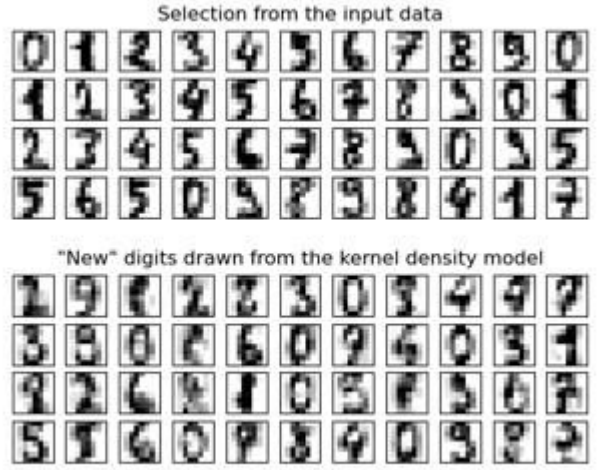
Aprendizaje semisupervisado

El aprendizaje semisupervisado es un caso mixto de los vistos anteriormente. En esta ocasión, en el conjunto de datos de partida existen elementos etiquetados y otros que no lo están.

Vamos a utilizar el dataset "digits", que consta de imágenes de 8x8 píxeles que representan dígitos del 0 al 9.

Divide los datos en conjuntos etiquetados y no etiquetados y entrena dos modelos, uno supervisado y otro semisupervisado.

Después calculamos la precisión de ambos modelos clasificando los dígitos.

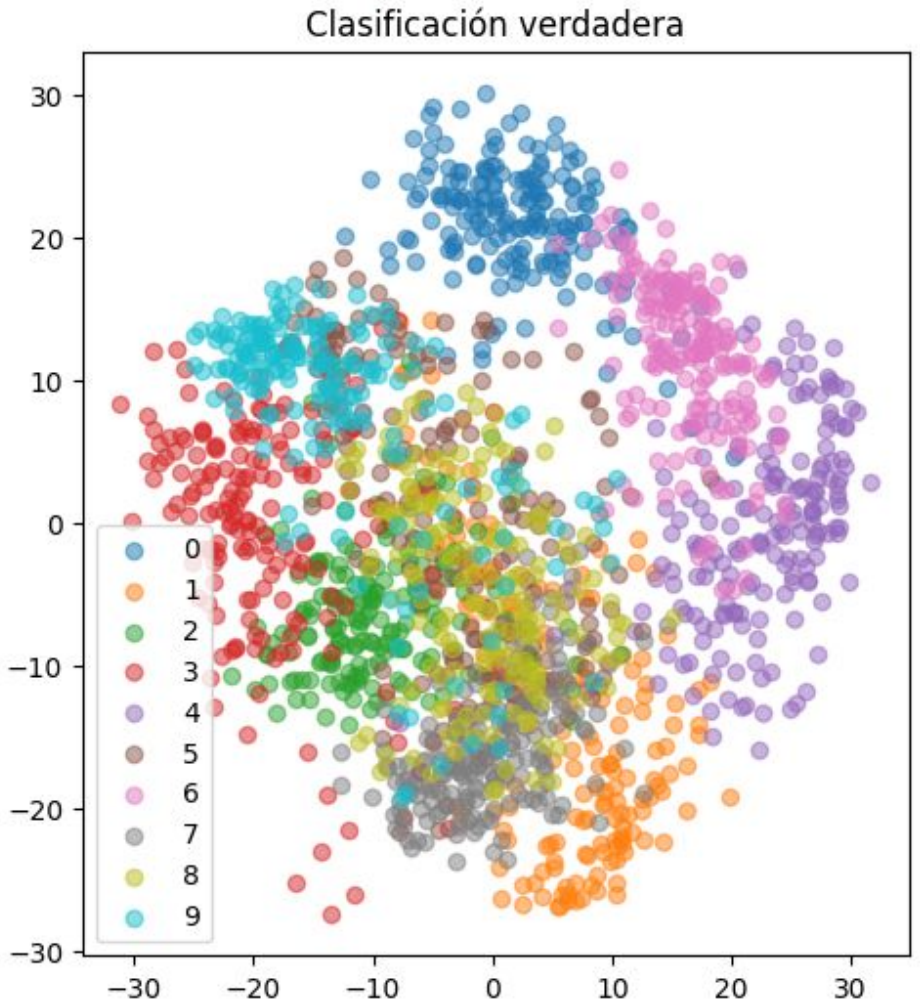


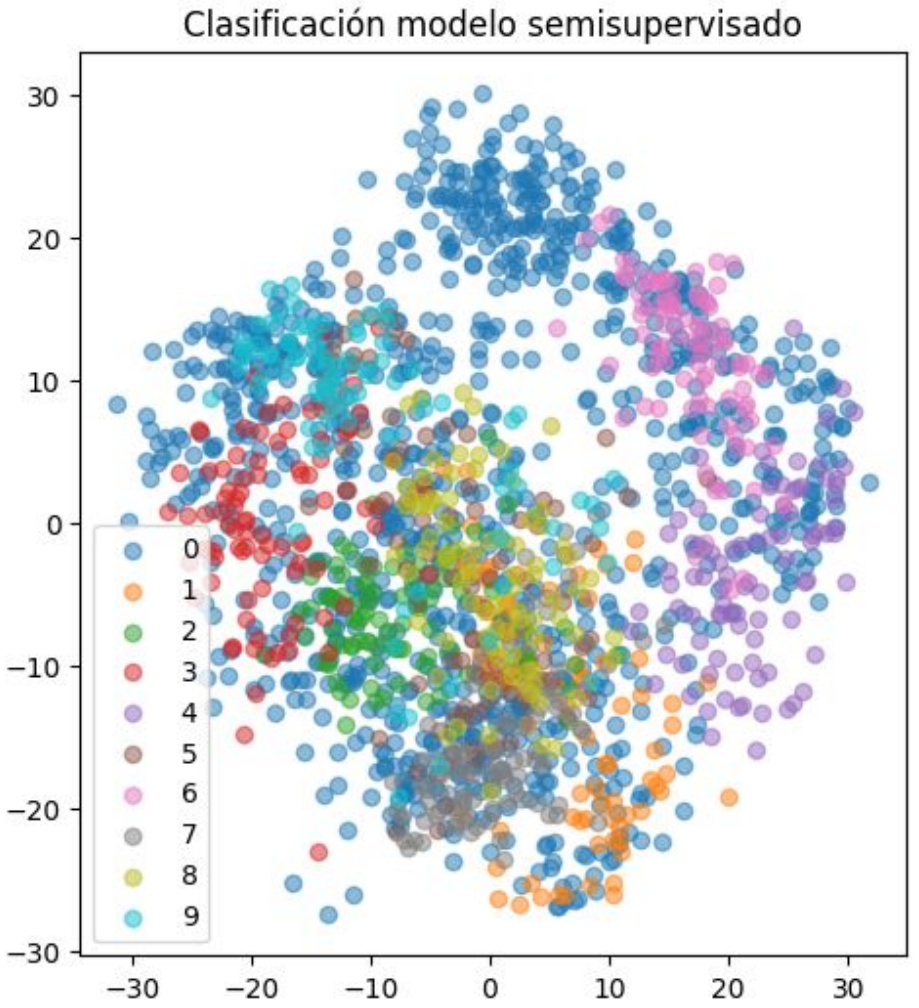
https://archive.ics.uci.edu/dataset/80/optical+recognition+of+handwritten+digits

```
from sklearn import datasets
from sklearn.semi_supervised import LabelSpreading
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

# Cargar el conjunto de datos de dígitos
digits = datasets.load_digits()
X = digits.data
y = digits.target

# Dividir los datos en conjunto etiquetado y no etiquetado
X_labeled, X_unlabeled, y_labeled, y_unlabeled = train_test_split(X,
y, test_size=0.8, random_state=42)

# Determinar la cantidad de datos a marcar como no etiquetados
porcentaje_no_etiquetados = 0.5  # Por ejemplo, se marcará el 50% de
los datos como no etiquetados
num_no_etiquetados = int(len(y) * porcentaje_no_etiquetados)

# Inicializar y entrenar un modelo supervisado (Logistic Regression)
supervised_model = LogisticRegression(max_iter=10000)
supervised_model.fit(X_labeled, y_labeled)

# Calcular la precisión del modelo supervisado
accuracy_supervised = supervised_model.score(X_labeled, y_labeled)
print(f"Precisión del modelo supervisado: {accuracy_supervised}")

# Crear un conjunto de datos con datos etiquetados y no etiquetados
para el modelo semisupervisado
X_train = X.copy()
y_train = y.copy()
y_train[-num_no_etiquetados:] = -1  # Marcar el último
'num_no_etiquetados'

# Inicializar y entrenar el modelo de propagación de
etiquetas
label_prop_model = LabelSpreading()
label_prop_model.fit(X_train, y_train)

# Predecir en el conjunto de datos completo
predicted_labels = label_prop_model.transduction_

# Calcular la precisión del modelo semisupervisado
accuracy_semisupervised = accuracy_score(y,
predicted_labels)
print(f"Precisión del modelo semisupervisado:
{accuracy_semisupervised}")

# Reducir la dimensionalidad a 2 dimensiones para
visualización
pca = PCA(n_components=2)
X_r = pca.fit_transform(X)

# Visualizar los resultados
plt.figure(figsize=(12, 6))
# Gráfico para el modelo supervisado
plt.subplot(1, 2, 1)
for i in range(10):
    plt.scatter(X_r[y == i, 0], X_r[y == i, 1],
label=str(i), alpha=0.5)
plt.title('Clasificación verdadera')
plt.legend()
# Gráfico para el modelo semisupervisado
plt.subplot(1, 2, 2)
for i in range(10):
    plt.scatter(X_r[predicted_labels == i, 0],
X_r[predicted_labels == i, 1], label=str(i), alpha=0.5)
plt.title('Clasificación modelo semisupervisado')
plt.legend()

plt.show()
```

Resultados

```
Precisión del modelo supervisado: 1.0
Precisión del modelo semisupervisado: 0.5492487479131887
```





Actividad a entregar:

Entrega un documento en PDF con las capturas de las pruebas que has realizado en cada uno de los modelos de aprendizaje. Explica los resultados e interpreta las gráficas.

Modifica las entradas de cada uno de los ejemplos y observa los resultados. Comenta tus impresiones.

Estudia las librerías utilizadas en Python.

En la práctica del aprendizaje semisupervisado, prueba a cambiar el % de entradas sin etiquetar y observa los resultados. ¿Cuántas entradas debemos etiquetar para superar una precisión del 80%?

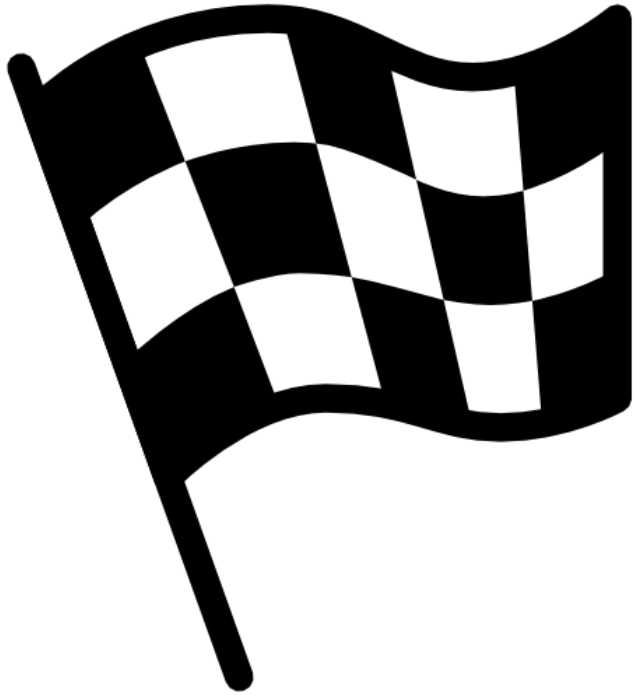
¿Para qué puedes utilizar los distintos modelos de aprendizaje estudiados en tu trabajo?

Selecciona uno de los códigos, y comenta línea a línea lo que hace.

¿Qué has aprendido?

Las diferencias entre:

- Aprendizaje automático supervisado
  - Regresión
  - Clasificación
- Aprendizaje no supervisado
  - Clustering
- Aprendizaje semisupervisado



"Con mucha diferencia, el mayor peligro de la Inteligencia Artificial es que las personas concluyen demasiado pronto que la entienden.".

Eliezer Yudkowsky, Investigador estadounidense de la Inteligencia Artificial y fundador de MIRI