guayerd

Fundamentos IA

Machine Learning

Clase 11

En colaboración con

IBM **SkillsBuild**

¡Bienvenidos!

¿Nos presentamos?

- ¿Qué recuerdan de la clase anterior?
- ¿Qué esperan aprender?
- ¿Tienen alguna pregunta?

guayerd

En colaboración con

IBM **SkillsBuild**

Contenidos

Por temas

09
- Demo

10
- Fundamentos

11
- Scikit-learn
- Demo asincrónica

12
- Introducción al entorno
- Modelado y relaciones

guayerd

En colaboración con

IBM **SkillsBuild**

Objetivos de la clase

- Preparación datos
- División train/test
- Proceso entrenamiento
- Evaluación modelos
- Algoritmos específicos

guayerd

En colaboración con

IBM **SkillsBuild**

Machine Learning

Modelado con scikit-learn

guayerd

En colaboración con

IBM **SkillsBuild**

Plataforma Skill Build: Machine Learning

Plan de formación

Python for Machine Learning: Unlocking the Power of Artificial Intelligence

Aproximadamente 23 horas · 281 · ★★★★★ 18

eLearning

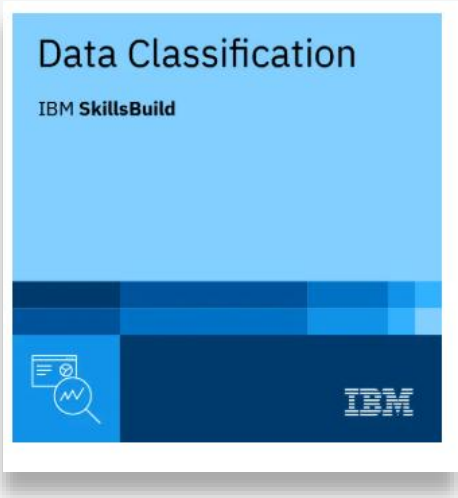
Modelos de IBM Granite para el desarrollo de softwar

1 hora 30 minutos · 8.825 · ★★★★★ 623

eLearning

Clasificación de datos con IBM Granit

1 hora 30 minutos · 10.885 · ★★★★★ 597



guayerd

En colaboración con

IBM **SkillsBuild**

¿Qué es Scikit-learn?

Biblioteca de Python para machine learning.

- Algoritmos listos para usar
- Integración con NumPy y Pandas
- Herramientas de evaluación y validación

Comandos

- **Instalación:** pip install scikit-learn
- **Uso:** from sklearn import [módulo]

Del proceso al código

El flujo en scikit-learn incluye los siguientes pasos:

- **División:** train_test_split
- **Entrenamiento:** fit()
- **Predicción:** predict()
- **Evaluación:** metrics

División

Para evaluar un modelo de forma confiable, separamos los datos en dos conjuntos:

- **Train (70–80%):** entrena al modelo con los datos
- **Test (20–30%):** evalúa el modelo con datos no vistos

Comando

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

De esta manera comprobamos si el modelo generaliza y no solo memoriza

Entrenamiento

El modelo aprende patrones a partir de los datos de entrenamiento.

- Usa las variables de entrada (X_train) y las etiquetas (y_train)
- Ajusta los parámetros internos para minimizar el error
- Este paso se realiza con el método fit()

Comando para regresión lineal

```
from sklearn.linear_model import LinearRegression
modelo = LinearRegression()
modelo.fit(X_train, y_train)
```

Comando para clasificación binaria

```
from sklearn.linear_model import LogisticRegression
modelo = LogisticRegression()
modelo.fit(X_train, y_train)
```

Predicciones

El modelo aplica lo aprendido para generar resultados sobre datos no vistos.

- Usa el conjunto de prueba (X_test)
- Devuelve valores numéricos (regresión) o clases/probabilidades (clasificación)
- Se utiliza el método predict()

Comando

```
# Regresión lineal
predicciones = modelo.predict(X_test)

# Clasificación binaria
predicciones = modelo.predict(X_test)
probabilidades = modelo.predict_proba(X_test)
```

Evaluación

El modelo se valida comparando las predicciones con los valores reales.

- **Accuracy:** Porcentaje de predicciones correctas
- **Matriz de confusión:** tabla que muestra aciertos y errores en predicciones
- **Error promedio:** distancia entre predicciones y valores reales

Accuracy

Porcentaje de **predicciones correctas sobre el total**.

- Se aplica en clasificación
- Mide cuántas veces acertó el modelo
- Valor entre 0 y 1 (más cercano a 1, mejor)

Comando

```
accuracy_score(y_test, y_pred)
```

Matriz de confusión

Tabla que **resume los aciertos y errores** de clasificación.

- Filas = valores reales
- Columnas = predicciones del modelo
- Permite ver qué clases confunde

Comando

```
confusion_matrix(y_test, y_pred)
```

Error promedio (MSE)

Mide la diferencia media entre valores reales y predichos.

- Se aplica en regresión
- Indica qué tan lejos están las predicciones
- Cuanto más bajo, mejor

Comando

```
mean_squared_error(y_test, y_pred)
```

Interpretando resultados

- Calcular predicciones correctas
- Evaluar detección de clientes premium (valor 1)
- Explicar el valor 50 en la matriz

Accuracy: 0.92		
Matriz de confusión	Pred 0	Pred 1
Real 0:	850	50
Real 1:	30	70

Visualización de resultados

Las métricas se pueden representar gráficamente para interpretar mejor el rendimiento.

- **Heatmap de matriz de confusión:** muestra aciertos y errores en una tabla de colores.
- **Barras de métricas:** compara valores como accuracy y error promedio.

Comando heatmap

```
matriz = confusion_matrix(y_test, predicciones)

sns.heatmap(matriz, annot=True, fmt="d")
```

Comando barras

```
plt.bar(['Accuracy','Error'], [accuracy, error_promedio])

plt.show()
```

Pipeline

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,
confusion_matrix, mean_squared_error

# División
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3)

# Entrenamiento
modelo = LogisticRegression()
modelo.fit(X_train, y_train)
```

```
# Predicciones
y_pred = modelo.predict(X_test)

# Evaluación
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Matriz de confusión:\n", confusion_matrix(y_test,
y_pred))
print("Error promedio:", mean_squared_error(y_test,
y_pred))
```

Venta de ropa online

En relación al caso visto la clase anterior:

1. Preparar los datos
2. Dividir en train/test
3. Entrenar modelo de clasificación
4. Accuracy, matriz de confusión y heatmap
5. Entrenar modelo de regresión
6. Calcular error promedio

visitas	fuente	dispositivo	desc	items	tiempo	carrito	compra	importe
3	ads	mob	1	6	5	1	1	120
1	org	desk	0	2	1	0	0	0
4	email	mob	1	5	7	1	1	90
2	org	mob	0	3	3	0	0	0
5	ads	desk	0	7	8	1	1	150
1	email	desk	0	1	2	1	0	0
3	org	mob	1	4	4	0	0	0
6	ads	mob	1	9	10	1	1	210
2	email	desk	0	3	3	0	0	0
4	org	desk	1	5	6	1	1	110
2	ads	mob	0	2	3	0	0	0
5	email	desk	1	8	9	1	1	180

Proyecto

Tienda Aurelion

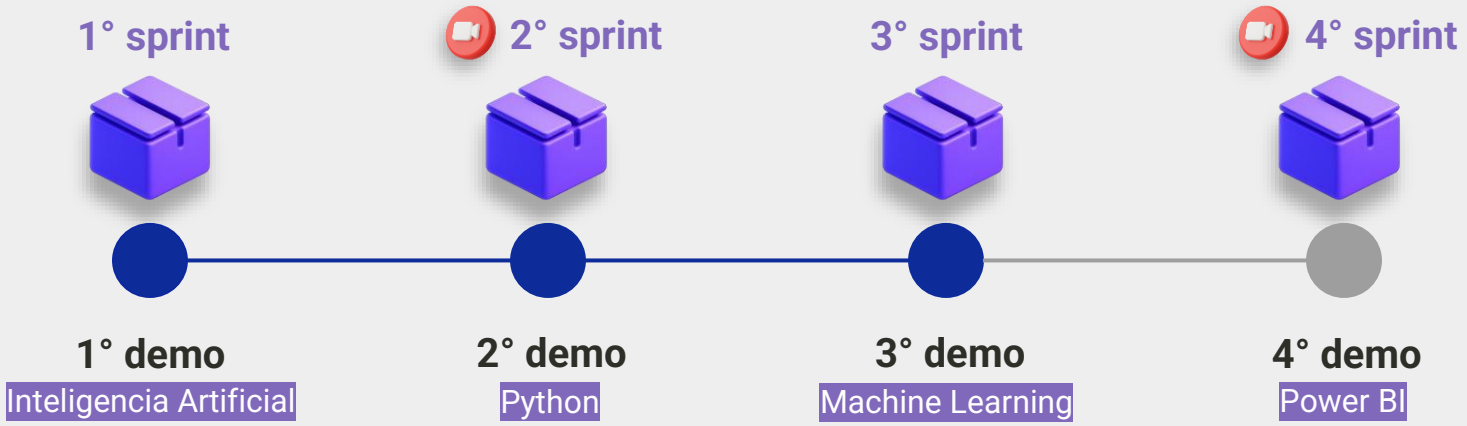
- **Documentación:** notebook Markdown
- **Desarrollo técnico:** programa Python
- **Visualización de datos:** dashboard en Power BI
- **Presentación oral:** problema, solución y hallazgos

guayerd

En colaboración con

IBM **SkillsBuild**

Sprints Project



guayerd

En colaboración con

IBM **SkillsBuild**

Características

3° demo: asincrónica

- **Tema:** Machine Learning
- **Medio:** Carpeta personal en Drive
- **Fecha límite:** 02/11
- **Devolución:** En salitas por equipo

Contenido

3º demo: asincrónica

Documentación actualizada (.md)

- Objetivo (predecir o clasificar)
- Algoritmo elegido y justificación
- Entradas (X) y salida (y)
- Métricas de evaluación
- Modelo ML implementado
- División train/test y entrenamiento
- Predicciones y métricas calculadas
- Resultados en uno o más gráficos

Programa actualizado (.py)

- Información actualizada del proyecto

Implementación ML

Trabajo en equipo

En relación a la base de datos.

1. Incluye un **modelo** ML (regresión o clasificación)
2. Divide en **train/test** y **entrena** el modelo
3. Genera **predicciones** y calcula **métricas** básicas
4. Representa resultados en uno o más **gráficos**

Retro

¿Cómo nos vamos?

- ¿Qué fue lo más útil de la clase?
- ¿Qué parte te costó más?
- ¿Qué te gustaría repasar o reforzar?

guayerd

En colaboración con

IBM **SkillsBuild**