



Minería de Datos en Python

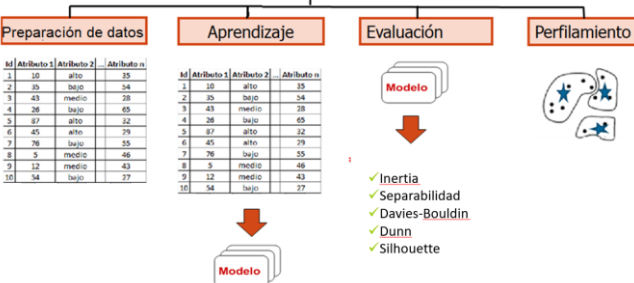
PhD. Ana Isabel Oviedo
<ana.oviedo@upb.edu.co>

UNIVERSIDAD PONTIFICIA BOLIVARIANA
FACULTAD DE INGENIERÍA EN TIC
2026

Minería Descriptiva

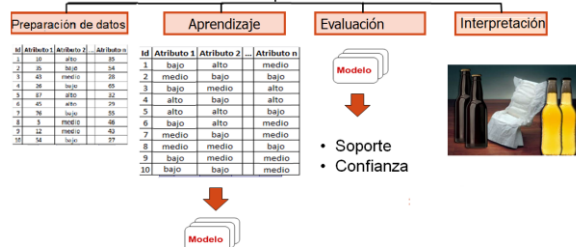
Clustering

Ciclo de Vida



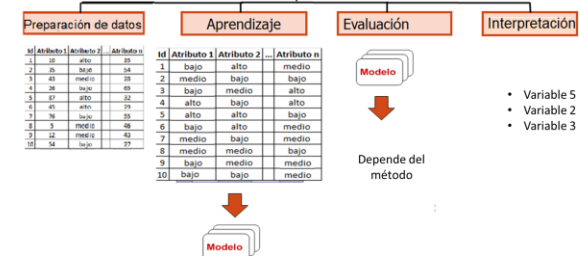
Asociación

Ciclo de Vida



Selección de Factores

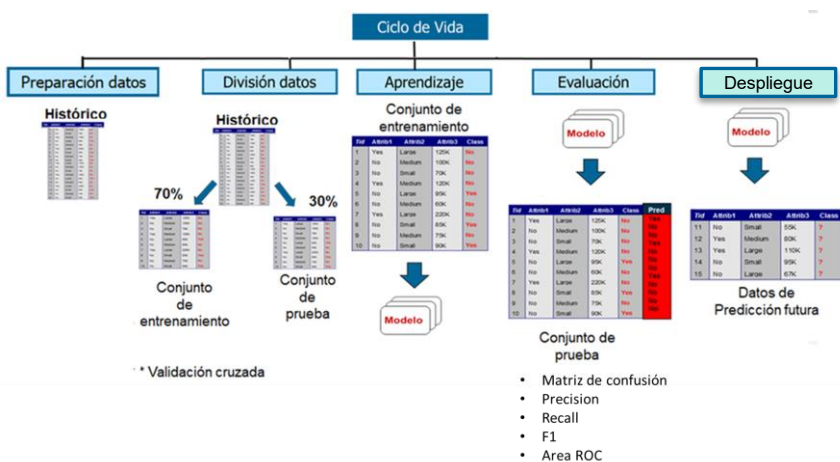
Ciclo de Vida



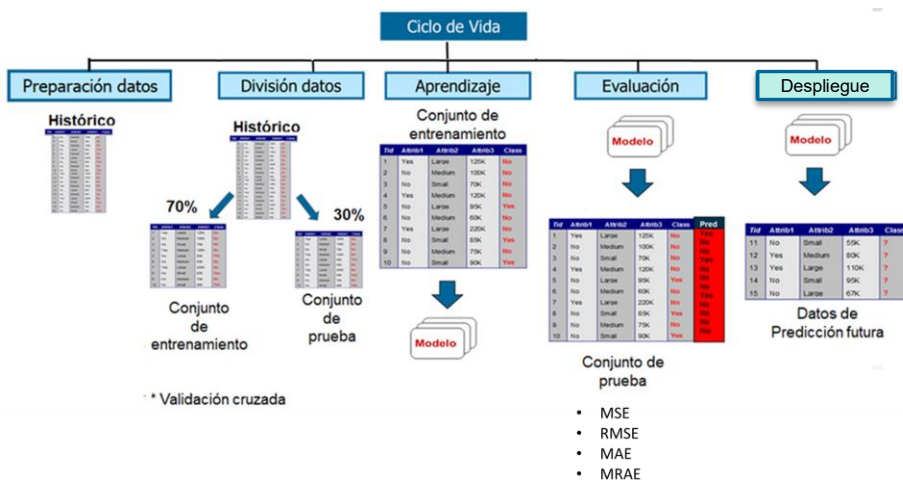
Análisis	Medidas de Evaluación	Algoritmo	Parámetros de Configuración	Código Python
Clustering: descubrimiento de similitudes	Inertia, separabilidad, silueta, Índice Dun, Índice DB	Kmeans	Cantidad de clusters, cantidad de iteraciones	<pre>from sklearn.cluster import KMeans model = KMeans(n_clusters=3, max_iter=100)</pre>
Reglas de Asociación: descubrimiento de co-ocurrencias (patrones repetidos)	Soporte y confianza	Apriori	Soporte mínimo y confianza mínima	<pre>from apyori import apriori reglas=apriori(data.values, min_support=0.1, min_confidence=0.6) reglas=list(reglas)</pre>
Selección de Factores: descubrimiento de variables relevantes	* Correlaciones: NA * Arbol de clasificación/ Regresión logística: matriz de confusión * Arbol de regresión/ Regresión lineal: medida de error	Correlation Arbol de decisión Regresión lineal/logística	Correlation: NA Arbol: criterio, cantidad de registros en las hojas, profundidad Regresión: si se desea normalizar, si es multiclase	<pre>data.corr() from sklearn.tree import DecisionTreeClassifier model = DecisionTreeClassifier(criterion='gini', min_samples_leaf=2, max_depth=10) from sklearn.linear_model import LogisticRegression model = LogisticRegression(solver='liblinear', multi_class='auto')</pre>

Minería Predictiva

Clasificación



Regresión



Minería Predictiva

Análisis	Medidas de Evaluación	Algoritmo	Parámetros de Configuración	Código Python
Clasificación: predicción de una variable objetivo categórica	Matriz de confusión, precisión, recall, exactitud, f1, area ROC	Arboles de Clasificación	Índice de información (entropy, gini), mínima cantidad de datos en las hojas, máxima profundidad	<pre>from sklearn.tree import DecisionTreeClassifier model = DecisionTreeClassifier(criterion='gini', min_samples_leaf=2, max_depth=10)</pre>
		KNN	Cantidad de vecinos, medida de distancia	<pre>from sklearn.neighbors import KNeighborsClassifier modelKnn = KNeighborsClassifier(n_neighbors=1, metric='euclidean')</pre>
		Redes Neuronales	Cantidad de capas y neuronas, función de activación, tasa de aprendizaje, valor inicial de la tasa de aprendizaje, momentum, cantidad de iteraciones	<pre>from sklearn.neural_network import MLPClassifier modelRN = MLPClassifier(activation="logistic", hidden_layer_sizes=(4), learning_rate='constant', learning_rate_init=0.2, momentum= 0.3, max_iter=500)</pre>
		SVM	kernel	<pre>from sklearn.svm import SVC modelSVM = SVC(kernel='linear') # 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'</pre>
		Otros: Regresión logística, bayesiano		
Regresión: Predicción de una variable objetivo numérica	(MSE) – Error cuadrático medio (RMSE) -Raíz del error cuadrático medio (MAE) -Error medio absoluto (R²) – R cuadrado (MRAE) – Error relativo absoluto	Arboles de Regresión	Índice de información (mse, mae), mínima cantidad de datos en las hojas, máxima profundidad	<pre>from sklearn.tree import DecisionTreeRegressor model_Tree = DecisionTreeRegressor(criterion='mse', min_samples_leaf=2, max_depth=None)</pre>
		KNN para Regresión	Cantidad de vecinos, medida de distancia	<pre>from sklearn.neighbors import KNeighborsRegressor model_Knn = KNeighborsRegressor(n_neighbors=1, metric='euclidean')</pre>
		Redes Neuronales	Cantidad de capas y neuronas, función de activación, tasa de aprendizaje, valor inicial de la tasa de aprendizaje, momentum, cantidad de iteraciones	<pre>from sklearn.neural_network import MLPRegressor model_RN = MLPRegressor(activation="logistic", hidden_layer_sizes=(4), learning_rate='constant', learning_rate_init=0.2, momentum= 0.3, max_iter=500)</pre>
		SVM para Regresión	kernel	<pre>from sklearn.svm import SVR model_SVM = SVR(kernel='linear') # 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'</pre>
		Otros: series de tiempo, regresión lineal		

Despliegue

Predicción de inversión en una tienda de videojuegos

Edad

14 20 52

Videojuego

'Mass Effect' ▼

Plataforma

'Play Station' ▼

Sexo

Hombre ▼

Consumidor_habitual

True ▼

Predicciones

	Edad	videojuego	Plataforma	Sexo	Consumidor_habitual	Tree	Knn	Prediccion_NN
0	20	'Mass Effect'	'Play Station'	Hombre	True	410	500	347.0949

Predicción de ventas

Sube el archivo CSV con los datos a predecir



Drag and drop file here

Limit 200MB per file • CSV

Browse files



ventas - despliegue.csv 233.0B



	videojuego	Edad	Sexo	Plataforma	Consumidor_habitual
0	'Mass Effect'	33	Hombre	Otros	<input type="checkbox"/>
1	'KOA: Reckoning'	30	Mujer	PC	<input checked="" type="checkbox"/>
2	'Dead Space'	22	Hombre	'Xbox'	<input type="checkbox"/>
3	'Crisis'	17	Hombre	'Play Station'	<input type="checkbox"/>
4	'Sim City'	27	Mujer	Otros	<input checked="" type="checkbox"/>

Predecir

Despliegue Github- Streamlit

1. Crea una cuenta en GitHub

- Ve a <https://github.com>
- Haz clic en Sign up
- Sigue los pasos para crear tu cuenta (usuario, correo, contraseña)

```
requirements.txt  
pandas  
numpy  
matplotlib  
scikit-learn
```

2. Crea un nuevo repositorio en GitHub

- Inicia sesión en GitHub
- Haz clic en el botón "+" en la esquina superior derecha > New repository
- Ponle un nombre, por ejemplo: mi-app-streamlit
- Marca la opción "Add a README file"
- Haz clic en Create repository

3. Sube tu proyecto a GitHub

- En tu repositorio, haz clic en Add file > Upload files
- Arrastra tus archivos (app.py, requirements.txt, modelo.pkl.). También puedes subir y trabajar colaborativamente los notebooks de creación de los modelos, pero ten cuidado si son datos confidenciales.
- Haz clic en Commit changes


Despliegue Github- Streamlit

4. Conecta tu proyecto con Streamlit Community Cloud

- Ve a <https://streamlit.io/cloud>
- Inicia sesión con tu cuenta de GitHub. Ve a Free/Login with Github
- Haz clic en "Create app"/Deploy a public app from Github
- Selecciona:
 - El repositorio (ej. tu-usuario/mi-app-streamlit)
 - La rama (generalmente main)
 - El archivo principal (app.py)
- Haz clic en Deploy

Despliegue Github- Streamlit

Para hacer actualizaciones

- Ve a tu repositorio en GitHub.
- Abre el archivo requirements.txt.
- Haz clic en el ícono de lápiz  (Editar).
- Baja hasta el final y haz clic en Commit changes.
- Ve a tu app en Streamlit Community Cloud.
- Usa el botón de menú → "Clear cache and restart"