Folha de Dicas: Construindo Modelos de Aprendizado Não Supervisionado

Modelos de aprendizado não supervisionado

Nome do Modelo	Descrição Breve	Sintaxe do Código
UMAP	UMAP (Uniform Manifold Approximation and Projection) é usado para redução de dimensionalidade. **Prós:** Alto desempenho, preserva a estrutura global. **Contras:** Sensível a parâmetros. **Aplicações:** Visualização de dados, extração de características. **Hiperparâmetros chave:** • **n_neighbors:** Controla o tamanho do bairro local (padrão = 15). • **min_dist:** Controla a distância mínima entre pontos no espaço embutido (padrão = 0.1). • **n_components:** A dimensionalidade da incorporação (padrão = 2).	`from umap.umap_ import UMAP` `umap = UMAP(n_neighbors=15, min_dist=0.1, n_components=2)`
t-SNE	t-SNE (t-Distributed Stochastic Neighbor Embedding) é uma técnica de redução de dimensionalidade não linear. **Prós:** Bom para visualizar dados de alta dimensionalidade. **Contras:** Computacionalmente caro, propenso a overfitting. **Aplicações:** Visualização de dados, detecção de anomalias. **Hiperparâmetros chave:** • **n_components:** O número de dimensões para a saída (padrão = 2). • **perplexity:** Equilibra a atenção entre aspectos locais e globais dos dados (padrão = 30). • **learning_rate:** Controla o tamanho do passo durante a otimização (padrão = 200).	`from sklearn.manifold import TSNE` `tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)`
PCA	PCA (análise de componentes principais) é usado para redução de dimensionalidade linear. **Prós:** Fácil de interpretar, reduz ruído. **Contras:** Linear, pode perder informações em dados não lineares. **Aplicações:** Extração de características, compressão. **Hiperparâmetros chave:** • **n_components:** Número de componentes principais a serem retidos (padrão = 2). • **whiten:** Se deve escalar os componentes (padrão = False). • **svd_solver:** O algoritmo para calcular os componentes (padrão = 'auto').	`from sklearn.decomposition import PCA` `pca = PCA(n_components=2)`
DBSCAN	DBSCAN (Density-Based Spatial Clustering of Applications with Noise) é um algoritmo de agrupamento baseado em densidade. **Prós:** Identifica outliers, não requer o número de clusters. **Contras:** Difícil com clusters de densidade variável. **Aplicações:** Detecção de anomalias, agrupamento de dados espaciais. **Hiperparâmetros chave:** • **eps:** A distância máxima entre dois pontos para serem considerados vizinhos (padrão = 0.5). • **min_samples:** Número mínimo de amostras em um bairro para formar um cluster (padrão = 5).	`from sklearn.cluster import DBSCAN` `dbscan = DBSCAN(eps=0.5, min_samples=5)`
HDBSCAN	HDBSCAN (Hierarchical DBSCAN) melhora o DBSCAN ao lidar com clusters de densidade variável. **Prós:** Melhor manejo de densidades variáveis. **Contras:** Pode ser mais lento que o DBSCAN. **Aplicações:** Grandes conjuntos de dados, problemas de agrupamento complexos. **Hiperparâmetros chave:** • **min_cluster_size:** O tamanho mínimo dos clusters (padrão = 5). • **min_samples:** Número mínimo de amostras para formar um cluster (padrão = 10).	`import hdbscan` `clusterer = hdbscan.HDBSCAN(min_cluster_size=5)`

Nome do Modelo	Descrição Breve	Sintaxe do Código
Agrupamento K-Means	K-Means é um algoritmo de agrupamento baseado em centróides que agrupa dados em k clusters. **Prós:** Eficiente, simples de implementar. **Contras:** Sensível aos centróides iniciais dos clusters. **Aplicações:** Segmentação de clientes, reconhecimento de padrões. **Hiperparâmetros chave:** • **n_clusters:** Número de clusters (padrão = 8). • **init:** Método para inicializar os centróides ('k-means++' ou 'random', padrão = 'k-means++'). • **n_init:** Número de vezes que o algoritmo será executado com diferentes sementes de centróides (padrão = 10).	`from sklearn.cluster import KMeans` `kmeans = KMeans(n_clusters=3)`

Funções associadas usadas

Método	Descrição Breve	Sintaxe do Código
make_blobs	Gera blobs gaussianos isotrópicos para clustering.	`from sklearn.datasets import make_blobs` `X, y = make_blobs(n_samples=100, centers=2, random_state=42)`
multivariate_normal	Gera amostras de uma distribuição normal multivariada.	`from numpy.random import multivariate_normal` `samples = multivariate_normal(mean=[0, 0], cov=[[1, 0], [0, 1]], size=100)`

plotly.express.scatter_3d	Cria um gráfico de dispersão 3D usando Plotly Express.	`import plotly.express as px`
		`fig = px.scatter_3d(df, x='x', y='y', z='z')`
		`fig.show()`

geopandas.GeoDataFrame	Cria um GeoDataFrame a partir de um DataFrame Pandas.	`import geopandas as gpd` `gdf = gpd.GeoDataFrame(df, geometry='geometry')`
geopandas.to_crs	Transforma o sistema de referência de coordenadas de um GeoDataFrame.	`gdf = gdf.to_crs(epsg=3857)`

contextily.add_basemap	Adiciona um mapa base a um gráfico de GeoDataFrame para contexto.	`import contextily as ctx`
		`ax = gdf.plot(figsize=(10, 10))`
		`ctx.add_basemap(ax)`

pca.explained_variance_ratio_	Retorna a proporção da variância explicada por cada componente principal.	`from sklearn.decomposition import PCA`
		`pca = PCA(n_components=2)`
		`pca.fit(X)`
		`variance_ratio = pca.explained_variance_ratio_`

Autor

Jeff Grossman
Abhishek Gagneja