Aprendizaje Supervisado en Machine Learning

Double-click or double-tap this to edit

El aprendizaje supervisado es una técnica de *machine learning* en la que un modelo aprende a partir de datos etiquetados para hacer predicciones. Se subdivide en dos categorías principales: algoritmos de regresión y algoritmos de clasificación.

Algoritmos de Regresión

Los algoritmos de regresión son utilizados para predecir valores continuos. Los más comunes incluyen:

- **Regresión lineal**: Ajusta una línea a los datos minimizando el error.
- **Árboles de decisión**: Modelos basados en estructuras jerárquicas que dividen el espacio de entrada en regiones homogéneas.
- **Random Forest**: Conjunto de múltiples árboles de decisión para mejorar la precisión y reducir la varianza.
- **Máquinas de soporte vectorial (SVM)**: Utilizan hiperplanos para separar los datos y pueden usarse para regresión mediante la técnica SVR.

Algoritmos de Clasificación

Estos algoritmos asignan datos a categorías específicas:

- **Regresión logística**: Se usa para problemas binarios al modelar la probabilidad de una clase.
- **Árboles de decisión**: Similares a los de regresión, pero usados para clasificación.
- **Random Forest**: Variante de los árboles de decisión con mejor generalización.
- **SVM**: Encuentra el mejor hiperplano que separa las clases en el espacio de entrada.
- **KNN (K-Nearest Neighbors)**: Clasifica una instancia según la mayoría de sus vecinos más cercanos.

Principales Parámetros

Para ajustar estos algoritmos en Python, se pueden modificar varios parámetros, como:

- **Regresión Lineal**: fit_intercept, normalize
- **Árboles de Decisión**: max_depth, min_samples_split, min_samples_leaf

- **Random Forest**: n_estimators, max_features
- **SVM**: C, kernel, gamma
- **KNN**: n_neighbors, weights

Como científicas de datos

- [ ] **Hiperparámetros**: Configuraciones previas al entrenamiento, como el número de árboles en Random Forest o el valor de C en SVM para tratar de encontrar los mejores modelos para nuestros datasets.
- [ ] **Validación cruzada**: Técnica para evaluar el modelo dividiendo los datos en múltiples conjuntos de entrenamiento y prueba.
- [ ] **Underfitting**: Cuando el modelo es demasiado simple y no captura la estructura de los datos.
- [ ] **Overfitting**: Cuando el modelo se ajusta demasiado a los datos de entrenamiento y tiene bajo rendimiento en datos nuevos.

Estrategias para Evitar el Overfitting

Para evitar el *overfitting* en modelos de *machine learning*, podemos aplicar varias estrategias:

1. **Regularización**
   Agrega una penalización a los coeficientes del modelo para evitar que se ajusten demasiado a los datos de entrenamiento. Algunas técnicas incluyen:
   - L1 (Lasso) y L2 (Ridge) en regresión lineal.
   - Parámetro C en SVM, que controla la tolerancia al error.
2. **Limitación de la Complejidad del Modelo**
   Reduce la capacidad del modelo para aprender patrones irrelevantes:
   - En árboles de decisión: ajusta max_depth, min_samples_split y min_samples_leaf.
   - En Random Forest: usa max_features para limitar el número de características analizadas por cada árbol.
3. **Aumento de Datos (Data Augmentation)**
   Si el conjunto de datos es pequeño, puedes expandirlo aplicando transformaciones como rotaciones, recortes o modificaciones en imágenes o texto.
4. **Uso de Validación Cruzada**
   Divide los datos en varios subconjuntos y entrena el modelo varias veces para evaluar su desempeño en diferentes combinaciones de datos.
5. **Eliminación de Características Irrelevantes**
   Reducir la cantidad de variables evita que el modelo aprenda ruido innecesario:
   - Usa Análisis de Componentes Principales (PCA) o algoritmos de selección de características.
6. **Uso de Dropout (para Redes Neuronales)**
   En redes neuronales profundas, Dropout desconecta aleatoriamente neuronas durante el entrenamiento para evitar la sobredependencia en ciertos patrones.
7. **Aumento del Conjunto de Datos**
   Si es posible, recolectar más datos de entrenamiento ayuda al modelo a aprender de manera más generalizada.

La clave es encontrar el equilibrio adecuado entre sesgo y varianza, para que el modelo generalice bien en datos nuevos sin perder precisión.