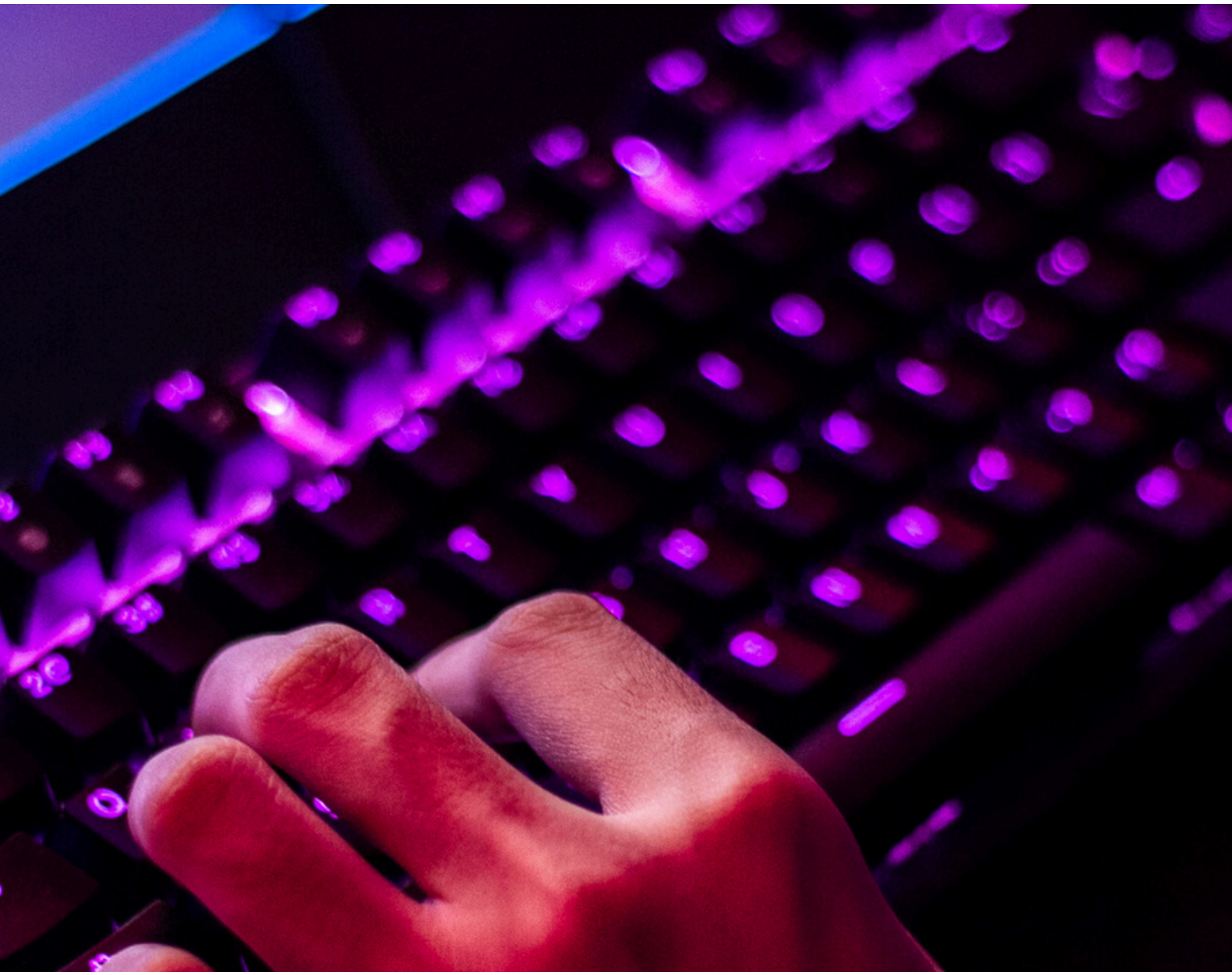


# Introducción a los modelos de aprendizaje automático

# Introducción a los modelos de aprendizaje automatico

En esta actividad, explicaremos los aspectos fundamentales para desarrollar modelos eficaces en inteligencia artificial. Esto proporcionará una sólida comprensión de los pasos necesarios para construir modelos de aprendizaje automático efectivos y escalables.

## 1. Selección del Algoritmo en Aprendizaje Automático

La selección del algoritmo adecuado es un paso crítico en el proceso de desarrollo de modelos de aprendizaje automático. Implica evaluar y elegir el algoritmo que mejor se adapte al problema específico que estamos tratando de resolver y a los datos disponibles.

- **Comprender el Problema:**

Antes de seleccionar un algoritmo, es crucial comprender completamente el problema que estamos abordando. Esto incluye definir claramente los objetivos del proyecto, entender el tipo de datos involucrados y tener una idea clara de las métricas de evaluación que se utilizarán para medir el rendimiento del modelo.

- **Conocimiento de los Algoritmos Disponibles:**

Es importante tener un conocimiento profundo de los diferentes tipos de algoritmos de aprendizaje automático disponibles, como algoritmos de regresión, clasificación, agrupamiento, y aprendizaje no supervisado, entre otros. Cada algoritmo tiene sus propias características, suposiciones y aplicaciones específicas.

- **Evaluación de Algoritmos:**

Una vez que entendemos el problema y conocemos los algoritmos disponibles, debemos evaluar cómo se adaptan a nuestro caso particular. Esto implica considerar factores como la naturaleza de los datos (numéricos, categóricos, textuales), el tamaño del conjunto de datos, la dimensionalidad, la presencia de outliers, la interpretabilidad del modelo, entre otros.

- **Experimentación y Comparación:**

Es útil realizar experimentos con varios algoritmos y comparar sus resultados utilizando técnicas de validación cruzada y métricas de evaluación relevantes. Esto nos ayudará a identificar cuál es el algoritmo que produce los mejores resultados para nuestro problema específico.

- **Ajuste y Optimización:**

Una vez seleccionado el algoritmo inicial, es posible que sea necesario ajustar y optimizar sus parámetros para mejorar su rendimiento. Esto se puede hacer utilizando técnicas como la búsqueda de hiperparámetros o el ajuste fino del modelo.

- **Reevaluación Constante:**

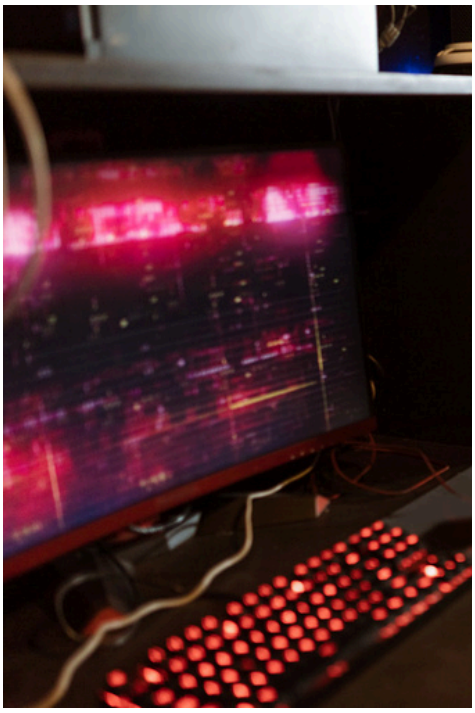
La elección del algoritmo no es una decisión única y permanente. A medida que se obtienen más datos o se comprende mejor el problema, es importante reevaluar la elección del algoritmo y estar dispuesto a cambiar si es necesario para obtener mejores resultados.

# 2. Preparación de Datos en Machine Learning

La preparación de datos es un paso crucial en el desarrollo de modelos de Machine Learning, ya que garantiza que los datos estén en un formato adecuado para el entrenamiento del modelo y que se puedan extraer características relevantes para mejorar el rendimiento del modelo. Veamos algunos aspectos clave de la preparación de datos:

## Asegurar que los Datos estén en Formato Adecuado

Antes de comenzar el proceso de entrenamiento del modelo, es fundamental asegurarse de que los datos estén en un formato adecuado. Esto implica realizar las siguientes tareas:



- **Manejo de Datos Faltantes:** Identificar y manejar los valores faltantes en el conjunto de datos. Esto puede implicar eliminar las filas o columnas con valores faltantes, imputar valores utilizando técnicas como la media, la mediana o la moda, o utilizar modelos predictivos para estimar los valores faltantes.

## Codificación de Variables Categóricas

Convertir las variables categóricas en un formato numérico que pueda ser interpretado por los algoritmos de Machine Learning. Esto puede hacerse utilizando técnicas como la codificación one-hot o el etiquetado de categorías.

## Escalado de Características

Escalar las características numéricas para que estén en la misma escala. Esto es especialmente importante para algoritmos basados en distancias, como el descenso de gradiente estocástico o los algoritmos de vecinos más cercanos.

## Modelos de Machine Learning

Existen varias técnicas de preprocesamiento específicas que pueden mejorar el rendimiento de los modelos de Machine Learning como:

| Reducción de Dimensionalidad | Tratamiento de Outliers |
|---|---|
| Utilizar técnicas como el análisis de componentes principales (PCA) o la selección de características para reducir la dimensionalidad de los datos y eliminar características irrelevantes o redundantes. | Identificar y manejar valores atípicos que pueden afectar negativamente el rendimiento del modelo. Esto puede implicar eliminar los outliers, transformarlos o asignarles valores específicos. |
| **Normalización y Estandarización** | **Generación de Características** |
| Normalizar o estandarizar las características numéricas para que tengan una media de cero y una desviación estándar de uno. Esto puede ayudar a mejorar la convergencia de los algoritmos de optimización y el rendimiento del modelo. | Crear nuevas características a partir de las características existentes que puedan capturar mejor la estructura subyacente de los datos y mejorar la capacidad predictiva del modelo. |

# 4. Entrenamiento y Ajuste del Modelo en Aprendizaje Automático

El proceso de entrenamiento y ajuste del modelo es fundamental en el desarrollo de sistemas de inteligencia artificial efectivos.

- **Implementación Práctica del Proceso de Entrenamiento:**

**Selección del Modelo**

Primero, debemos elegir el modelo de Machine Learning que se ajuste mejor al problema en cuestión. Scikit-Learn proporciona una amplia gama de modelos predefinidos para diferentes tareas, como regresión, clasificación, agrupamiento, entre otros.

**División del Conjunto de Datos**

Dividimos el conjunto de datos en conjuntos de entrenamiento y prueba. El conjunto de entrenamiento se utiliza para entrenar el modelo, mientras que el conjunto de prueba se utiliza para evaluar su rendimiento.

**Entrenamiento del Modelo**

Utilizamos el conjunto de entrenamiento para ajustar los parámetros del modelo y minimizar una función de pérdida o maximizar una métrica de rendimiento, como precisión o coeficiente de determinación. Esto se hace utilizando el método fit() del estimador en Scikit-Learn.

**Evaluación del Modelo**

Una vez entrenado el modelo, lo evaluamos utilizando el conjunto de prueba para medir su rendimiento en datos no vistos. Esto nos proporciona una estimación de la capacidad de generalización del modelo.

- **Ajuste de Hiperparámetros para Mejorar el Rendimiento:**

Los hiperparámetros son configuraciones que no se aprenden directamente del conjunto de datos, pero afectan el comportamiento y el rendimiento del modelo durante el entrenamiento. Ajustar los hiperparámetros adecuadamente puede mejorar significativamente el rendimiento del modelo. Scikit-Learn proporciona herramientas para realizar este ajuste, como la búsqueda de cuadrícula y la búsqueda aleatoria.

### Búsqueda de Cuadrícula

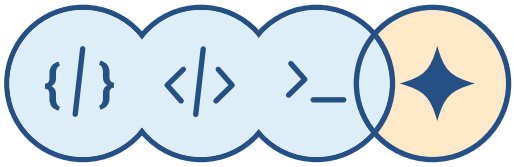
Consiste en definir una cuadrícula de valores para los hiperparámetros que queremos ajustar y evaluar el rendimiento del modelo para cada combinación de valores en la cuadrícula. Scikit-Learn proporciona la clase GridSearchCV para realizar esta búsqueda de cuadrícula de forma eficiente.

### Búsqueda Aleatoria

En lugar de evaluar todas las combinaciones posibles de hiperparámetros, la búsqueda aleatoria selecciona un conjunto aleatorio de combinaciones y evalúa su rendimiento. Esto puede ser más eficiente computacionalmente y puede encontrar configuraciones óptimas en menos tiempo.

- **Implementación Práctica con Scikit-Learn:**

A continuación se presenta un ejemplo de cómo implementar el proceso de entrenamiento y ajuste del modelo utilizando Scikit-Learn. Este código de Python entrena un modelo de clasificación utilizando el algoritmo Random Forest sobre un conjunto de datos generado sintéticamente utilizando la función make_moons de scikit-learn.



### Generación de los datos:

Se generan datos sintéticos utilizando la función make_moons de scikit-learn. Esta función genera un conjunto de datos bidimensional en forma de dos semicírculos (lunas) que se superponen.

```
X, y = make_moons(n_samples=100, noise=0.2)
```

### Visualización de los datos:

Se grafican los datos generados para visualizar la distribución de las dos clases.

```
plt.plot(X[:, 0][y==1], X[:, 1][y==1], "bs") # Clase 1
plt.plot(X[:, 0][y==0], X[:, 1][y==0], "g^") # Clase 2
plt.show()
```

## División del conjunto de datos:

Se divide el conjunto de datos en conjuntos de entrenamiento y prueba utilizando la función train_test_split de scikit-learn. Se reserva el 20% de los datos para pruebas.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

## Inicialización del modelo:

Se inicializa un clasificador Random Forest utilizando la clase RandomForestClassifier de scikit-learn.

```
model = RandomForestClassifier()
```

## Entrenamiento del modelo:

Se entrena el modelo de clasificación Random Forest utilizando los datos de entrenamiento.

```
model.fit(X_train, y_train)
```

## Evaluación del modelo:

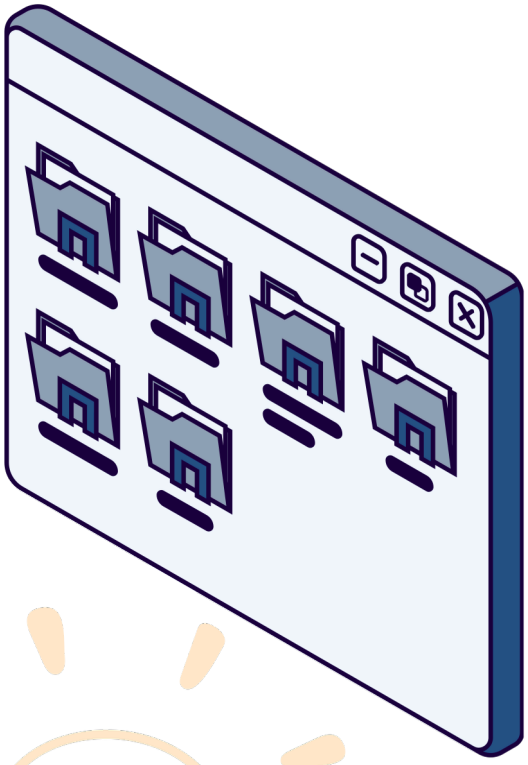
Se evalúa la precisión del modelo entrenado en los datos de prueba utilizando el método score.

```
accuracy = model.score(X_test, y_test)
print("Exactitud del modelo:", accuracy)
```

## Ajuste de hiperparámetros con búsqueda de cuadrícula

Se realiza una búsqueda de cuadrícula para encontrar los mejores hiperparámetros para el modelo utilizando la clase GridSearchCV de scikit-learn. Se prueban diferentes combinaciones de hiperparámetros definidas en el diccionario param_grid.

```
param_grid = {'n_estimators': [10, 20, 30],
              'max_depth': [None, 10, 20]}
grid_search = GridSearchCV(estimator=model,
param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)
```



Este código entrena un modelo de clasificación Random Forest sobre un conjunto de datos de lunas generados sintéticamente, evalúa su rendimiento en un conjunto de datos de prueba, y ajusta los hiperparámetros utilizando una búsqueda de cuadrícula para mejorar el rendimiento del modelo.