Evaluación de Clasificación

Métodos de evaluación de modelos

Son técnicas utilizadas para medir el rendimiento y la generalización de los modelos de aprendizaje automático. Esto incluye la validación cruzada, que consiste en dividir el conjunto de datos en múltiples subconjuntos para entrenar y evaluar el modelo repetidamente, y las curvas de aprendizaje, que muestran cómo cambia el rendimiento del modelo con respecto al tamaño del conjunto de datos de entrenamiento.

Interpretación de métricas de evaluación

Las métricas de evaluación son medidas utilizadas para interpretar el rendimiento del modelo. Entre las más comunes se encuentran:

La precisión

Mide la proporción de predicciones correctas.

El recall

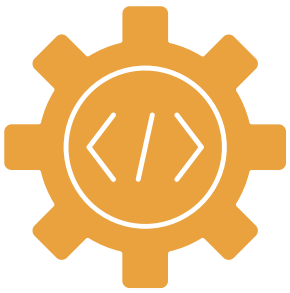
Mide la proporción de casos positivos que fueron correctamente identificados.

El F1-score

Es una combinación de precisión y recall.

Estas métricas son fundamentales para entender la capacidad predictiva y el comportamiento del modelo en diferentes situaciones.

Ejercicios prácticos de evaluación de modelos basados en árboles de decisión y Bosques Aleatorios



Los ejercicios prácticos de evaluación de modelos permiten aplicar los conceptos aprendidos en situaciones reales. Esto incluye la implementación de métricas de evaluación como precisión, recall y F1-score para evaluar modelos basados en árboles de decisión y Bosques Aleatorios. Además, se pueden realizar análisis comparativos entre diferentes modelos y ajustar los parámetros para optimizar el rendimiento del modelo.

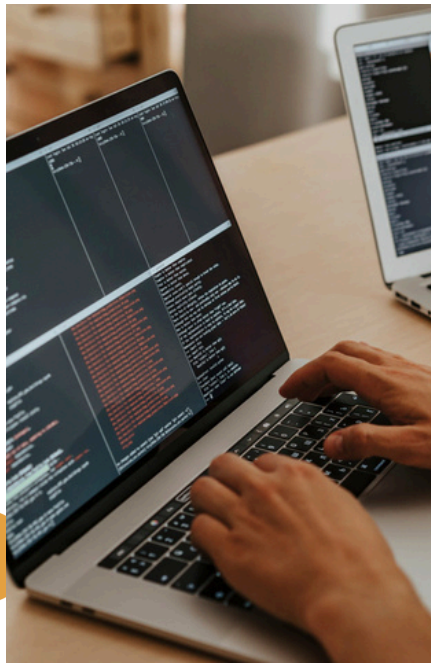
1. Precisión:

La precisión es una métrica que mide la proporción de predicciones positivas correctas entre todas las predicciones positivas realizadas por el modelo. Se calcula como el cociente entre los verdaderos positivos (TP) y la suma de los verdaderos positivos y los falsos positivos (FP).
La fórmula para calcular la precisión es:

Precisión: TP/(FP+TP)

2. Recall:

El recall, también conocido como sensibilidad o tasa de verdaderos positivos (TPR), mide la proporción de casos positivos que fueron correctamente identificados por el modelo. Se calcula como el cociente entre los verdaderos positivos (TP) y la suma de los verdaderos positivos y los falsos negativos (FN).
La fórmula para calcular el recall es:



$$Recall=TP/(TP+FN)$$

3. F1-score:

El F1-score es una medida que combina la precisión y el recall en un solo valor. Se calcula como la media armónica de la precisión y el recall, lo que proporciona una métrica equilibrada entre ambas.
La fórmula para calcular el F1-score es:

$$F1\text{-}score=2*Precisión*Recall/(Presición + Recall)$$

Implementación de Validación Cruzada

```python
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import make_classification

# Generación de datos ficticios
X, y = make_classification(n_samples=1000, n_features=20,
random_state=42)

# Inicialización del modelo
clf = DecisionTreeClassifier()

# Validación cruzada con 5 folds
scores = cross_val_score(clf, X, y, cv=5)
print("Accuracy:", scores.mean())
```

Curvas de Aprendizaje

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import make_classification

# Generación de datos ficticios
X, y = make_classification(n_samples=1000, n_features=20,
random_state=42)

# Inicialización del modelo
clf = DecisionTreeClassifier()
```

```
# Cálculo de las curvas de aprendizaje
train_sizes, train_scores, test_scores = learning_curve(
      clf, X, y, train_sizes=np.linspace(0.1, 1.0, 10), cv=5)

# Visualización de las curvas de aprendizaje
plt.figure()
plt.title("Curvas de Aprendizaje")
plt.xlabel("Tamaño del Conjunto de Entrenamiento")
plt.ylabel("Puntuación")
plt.grid()

train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                 train_scores_mean + train_scores_std, alpha=0.1,
                 color="r")
plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.1,
color="g")
plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
      label="Puntuación de Entrenamiento")
plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
      label="Puntuación de Validación")

plt.legend(loc="best")
plt.show()
```

Interpretación de Métricas de Evaluación

```python
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split

# Generación de datos ficticios
X, y = make_classification(n_samples=1000, n_features=20,
random_state=42)

# División de los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Inicialización del modelo
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)

# Predicciones sobre el conjunto de prueba
y_pred = clf.predict(X_test)

# Cálculo de métricas de evaluación
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Precisión:", precision)
print("Recall:", recall)
print("F1-score:", f1)
```

Los ejercicios pueden ser similares a otros, usando árboles de decisión y Bosques Aleatorios, respectivamente. Se pueden usar los mismos códigos cambiando **DecisionTreeClassifier** por **RandomForestClassifier**.

Matriz de Confusión

La matriz de confusión es una tabla que describe el rendimiento del modelo en términos de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN). Es una herramienta poderosa para entender cómo el modelo clasifica las muestras en diferentes clases y para identificar cualquier sesgo o errores que pueda tener.
Veamos las definiciones de estos conceptos:

Verdaderos Positivos (TP)	Predicciones correctas de ejemplos positivos.
Predicciones correctas de ejemplos negativos.	**Verdaderos Negativos (TN)**
Falsos Positivos (FP)	Predicciones incorrectas de ejemplos negativos como positivos.
Predicciones incorrectas de ejemplos positivos como negativos.	**Falsos Negativos (FN)**

Interpretar la matriz de confusión nos permite entender mejor cómo nuestro modelo está clasificando las muestras y nos proporciona información detallada sobre su rendimiento en diferentes clases.

Ejercicio: Cálculo de Matriz de confusión

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

# Cargar el conjunto de datos
digits = load_digits()
X, y = digits.data, digits.target

# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Entrenar el modelo
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Realizar predicciones
y_pred = model.predict(X_test)
# Calcular métricas de evaluación
confusion_matrix = confusion_matrix(y_test, y_pred)
# Mostrar las métricas
print("confusion_matrix:", confusion_matrix)
```