Profundiza más

Recurso de Profundización

Implementación de regresión logística en *sklearn*

A continuación, se muestra un ejemplo de implementación de regresión logística en *Python* mediante *sklearn*, se añadió un comentario en cada línea de código para mejorar el entendimiento.

```python
# Importamos las librerías necesarias
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

# Cargamos el dataset Iris
from sklearn.datasets import load_iris
iris = load_iris()

# Convertimos a DataFrame para facilitar el manejo
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['target'] = iris.target

# Seleccionamos solo dos clases para simplificar el problema (Clasificación binaria)
df = df[df['target'] != 2]  # Filtramos solo las clases 0 y 1

# Definimos las variables predictoras (X) y la variable objetivo (y)
X = df[iris.feature_names]
y = df['target']

# Dividimos los datos en entrenamiento y prueba (80% entrenamiento, 20% prueba)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normalizamos las características para mejorar el rendimiento del modelo
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Creamos el modelo de regresión logística
model = LogisticRegression()

# Entrenamos el modelo
model.fit(X_train_scaled, y_train)

# Realizamos las predicciones
y_pred = model.predict(X_test_scaled)

# Evaluamos el rendimiento del modelo
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Precisión del modelo: {accuracy:.4f}")
print("Matriz de confusión:")
print(conf_matrix)
```

✓ 39.4s

Profundiza más

Implementación de Kernelized Support Vector Machine in sklearn

A continuación se muestra un ejemplo de implementación de *kernelized support vector machine* en *Python* mediante *sklearn*, se añadió un comentario en cada línea de código para un mejor entendimiento.

```python
# Importamos las librerías necesarias
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix

# Cargamos el dataset Breast Cancer
cancer = load_breast_cancer()

# Convertimos a variables X (características) y y (objetivo)
X = cancer.data
y = cancer.target

# Dividimos los datos en entrenamiento y prueba (80% entrenamiento, 20% prueba)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normalizamos las características para mejorar el rendimiento del modelo
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Creamos el modelo SVM con kernel RBF
model = SVC(kernel='rbf', gamma='scale', C=1)

# Entrenamos el modelo
model.fit(X_train_scaled, y_train)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normalizamos las características para mejorar el rendimiento del modelo
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Creamos el modelo SVM con kernel RBF
model = SVC(kernel='rbf', gamma='scale', C=1)

# Entrenamos el modelo
model.fit(X_train_scaled, y_train)

# Realizamos las predicciones
y_pred = model.predict(X_test_scaled)

# Evaluamos el rendimiento del modelo
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Precisión del modelo: {accuracy:.4f}")
print("Matriz de confusión:")
print(conf_matrix)
```