Ejercicio práctico estructurado en un (Jupyter Notebook, Colab) para que los estudiantes puedan seguir paso a paso la implementación y evaluación de los modelos de clasificación usando el dataset Iris.

Ejercicio Práctico: Comparación de Modelos de Clasificación con el Dataset Iris

Objetivo

Implementar y comparar el rendimiento de distintos modelos de clasificación aplicados al dataset Iris, usando métricas de evaluación estándar.

Pre-requisitos

Antes de comenzar, asegúrate de tener instaladas las siguientes librerías en tu entorno de Anaconda:

```
pip install numpy pandas matplotlib seaborn scikit-learn
```

---

1. Configuración del Entorno y Carga de Librerías

Primero, importa las librerías necesarias:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Verifica las versiones de las librerías utilizadas:

```
import sys
import sklearn
import matplotlib as plt1
print(f"Python: {sys.version}")
print(f"NumPy: {np.__version__}")
print(f"Pandas: {pd.__version__}")
print(f"Matplotlib: {plt1.__version__}")
print(f"Seaborn: {sns.__version__}")
print(f"Scikit-learn: {sklearn.__version__}")
```

---

2. Carga y Exploración del Dataset

Carga el dataset Iris desde sklearn y conviértelo en un DataFrame de pandas:

```
# Cargar el dataset Iris
iris = datasets.load_iris()

# Convertir a DataFrame
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = iris.target

# Mapeo de etiquetas numéricas a nombres de especies
df['species'] = df['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})

# Mostrar las primeras filas
df.head()
```

2.1 Análisis Exploratorio

```
# Información general del dataset
df.info()

# Resumen estadístico de los datos numéricos
df.describe()

# Comprobación de valores nulos
print("Valores nulos en el dataset:\n", df.isnull().sum())

# Visualización de la distribución de las clases
```

```
sns.countplot(x='species', data=df, palette='viridis')
plt.title('Distribución de las especies en el dataset Iris')
plt.show()
```

2.2 Matriz de Correlación

```
df_encoded = df.copy()
df_encoded['species'] = df_encoded['species'].astype('category').cat.codes

plt.figure(figsize=(10,6))
sns.heatmap(df_encoded.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Matriz de Correlación")
plt.show()
```

---

3. Preparación de los Datos

3.1 División del Dataset en Entrenamiento y Prueba

```
# Separación de características (X) y variable objetivo (y)
X = df.drop(columns=['species'])
y = df['species']

# División en conjunto de entrenamiento y prueba (80%-20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)
```

3.2 Normalización de los Datos

Algunos modelos como SVM se benefician de la normalización.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

---

4. Implementación y Entrenamiento de los Modelos

```
Definimos una función para entrenar y evaluar cada modelo:
def train_and_evaluate_model(model, X_train, X_test, y_train, y_test, model_name):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Evaluación
    acc = accuracy_score(y_test, y_pred)
    print(f"Precisión de {model_name}: {acc:.4f}\n")

    # Matriz de confusión
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(5,4))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
xticklabels=iris.target_names, yticklabels=iris.target_names)
    plt.xlabel('Predicho')
    plt.ylabel('Real')
    plt.title(f'Matriz de Confusión - {model_name}')
    plt.show()

    # Reporte de Clasificación
    print(f"Reporte de Clasificación - {model_name}:\n",
classification_report(y_test, y_pred))

# Modelos
models = {
    "k-Nearest Neighbors": KNeighborsClassifier(n_neighbors=5),
    "Árbol de Decisión": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "SVM": SVC(kernel='linear', random_state=42)
}

# Entrenamiento y Evaluación
for name, model in models.items():
    train_and_evaluate_model(model, X_train_scaled, X_test_scaled, y_train, y_test,
name)
```

---

5. Comparación de los Modelos

5.1 Comparación de Precisión

```
accuracy_results = {}

for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    acc = accuracy_score(y_test, y_pred)
    accuracy_results[name] = acc

# Visualización de la precisión de los modelos
plt.figure(figsize=(8,5))
sns.barplot(x=list(accuracy_results.keys()), y=list(accuracy_results.values()), hue
= x=list(accuracy_results.keys()), palette='coolwarm')
plt.ylim(0.8, 1)
plt.ylabel('Precisión')
plt.title('Comparación de Precisión entre Modelos')
plt.xticks(rotation=45)
plt.show()
```

---

6. Conclusión

Los estudiantes deben analizar los resultados obtenidos y responder las siguientes preguntas en celdas de texto:

1. ¿Qué modelo obtuvo la mayor precisión? ¿Por qué crees que fue el mejor?
2. ¿Cómo se comparan los resultados en términos de la matriz de confusión?
3. ¿Qué ventajas y desventajas observas en cada modelo en base a los resultados obtenidos?
4. ¿Cómo influye la normalización de datos en el rendimiento del modelo?

---

7. Entrega

Los estudiantes deben entregar su notebook en el formato:

📂 NombreApellidos_M2_E1.ipynb

Debe incluir: ✔ Código completo y funcional
✔ Resultados de precisión y matrices de confusión
✔ Gráficos y visualizaciones
✔ Respuestas a las preguntas de análisis

Duración Estimada: 2 horas

Criterios de Evaluación

✅ Configuración del entorno y carga de librerías (10 puntos)
✅ Carga y exploración de datos (10 puntos)
✅ División de datos (10 puntos)
✅ Implementación y entrenamiento de modelos (30 puntos)
✅ Evaluación de modelos y visualización de resultados (30 puntos)
✅ Claridad y organización del código, buenas prácticas (10 puntos)