ADSO-GA04-Actividad de Aprendizaje Seis Análisis de Datos

Eddie Alejandro Arenas Vital

Juan Pablo Oviedo Herrera

Jhon Jairo Guzmán Caballero

SENA

Ciencia De Datos

2828523

Luis Fernando Sánchez

4 de Abril de 2025

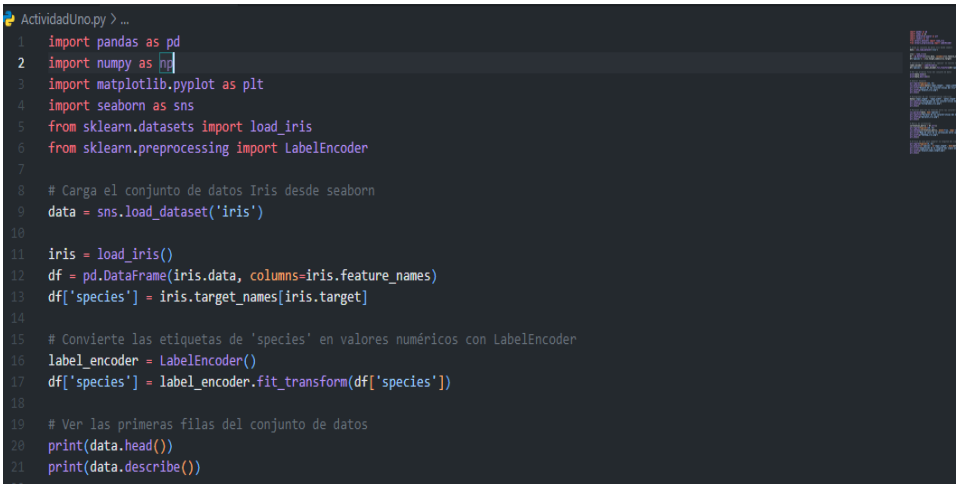
Análisis Exploratorio de Datos (EDA) - Conjunto de Datos Iris

1. Introducción

Este documento presenta un Análisis Exploratorio de Datos (EDA) utilizando el conjunto de datos Iris de la librería Seaborn. El objetivo es identificar patrones, valores atípicos y realizar visualizaciones básicas. Se usan las bibliotecas Pandas, Numpy, Matplotlib y Seaborn para el análisis y generación de gráficos.

2. Carga y descripción de los datos

Código para cargar y describir el conjunto de datos.



3. Identificación de valores atípicos

Los valores atípicos pueden afectar el análisis de datos. Para identificarlos, utilizamos diagramas de caja, mejor conocidos como boxplots.

```python
# Generar boxplots
plt.figure(figsize=(12, 8))
sns.boxplot(data=data[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']])
plt.title("Boxplot de las características del Iris")
plt.savefig("boxplot_iris.png")
plt.show()
```

4. Distribución de los datos

Se analizan las distribuciones de las variables mediante histogramas.

```python
# Histogramas de las características numéricas
data[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']].hist(bins=15, figsize=(12, 8))
plt.suptitle("Histogramas de las características del Iris")
plt.savefig("histogramas_iris.png")
plt.show()
```

5. Relación entre características

Se analizan relaciones entre características con gráficos de dispersión.

```python
# Pairplot para ver la relación entre las características numéricas
sns.pairplot(data, hue='species')
plt.suptitle("Pairplot de las características del Iris", y=1.02)
plt.savefig("pairplot_iris.png")
plt.show()
```

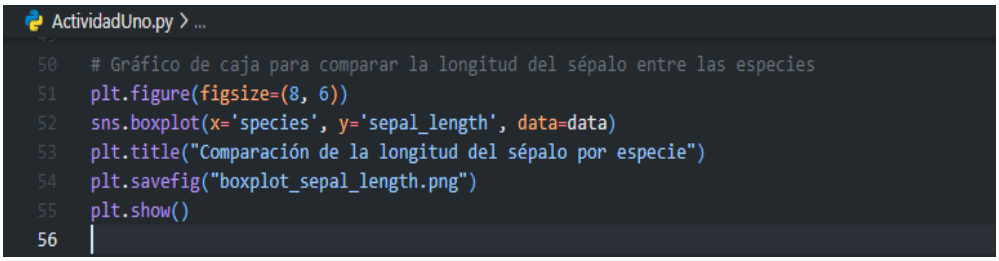
6. Correlación entre características

Se genera un mapa de calor para visualizar la correlación entre variables.

```python
# Matriz de correlación
correlation_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Mapa de calor de la correlación entre características")
plt.savefig("heatmap_iris.png")
plt.show()
```

7. Comparación entre especies

Se comparan las longitudes del sépalo entre especies con un gráfico de caja.



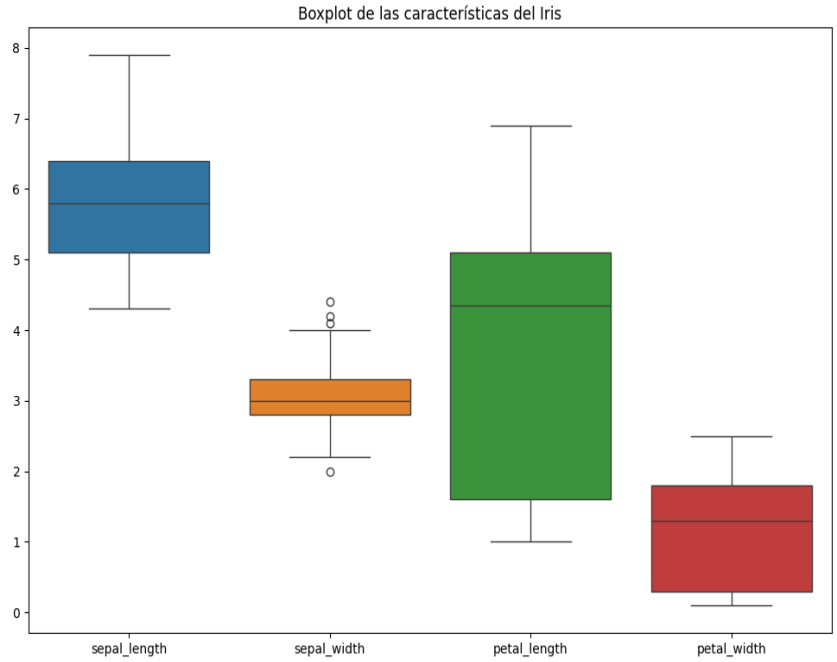
8. Conclusiones

El análisis del conjunto de datos "Iris" reveló diferencias significativas entre las especies en características como la longitud y el ancho del sépalo y pétalo, con Iris Setosa mostrando dimensiones más pequeñas. La visualización de datos a través de histogramas y scatter plots facilitó la identificación de patrones y correlaciones, anotando una alta correlación entre longitud del pétalo y longitud del sépalo. Se identificaron valores atípicos que requieren atención antes de implementar modelos predictivos.

9. Evidencias y Entrega

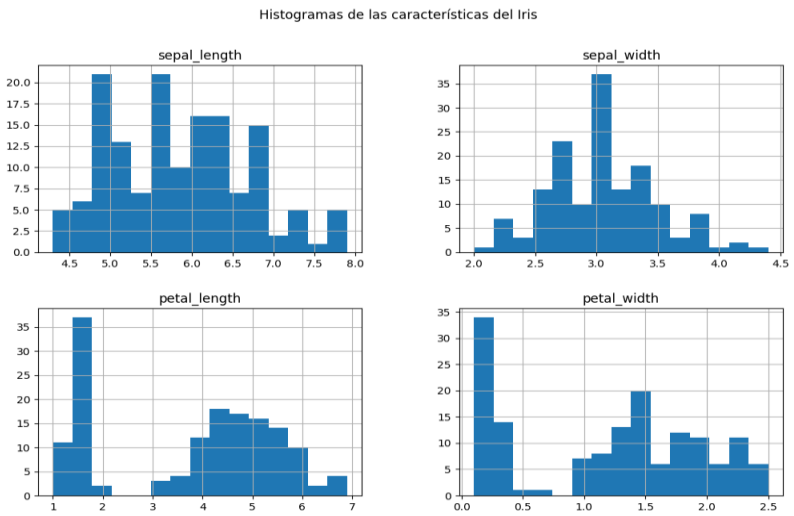
- Boxplot de las características del Iris

Este gráfico muestra la dispersión y posibles valores atípicos de las variables del conjunto de datos.



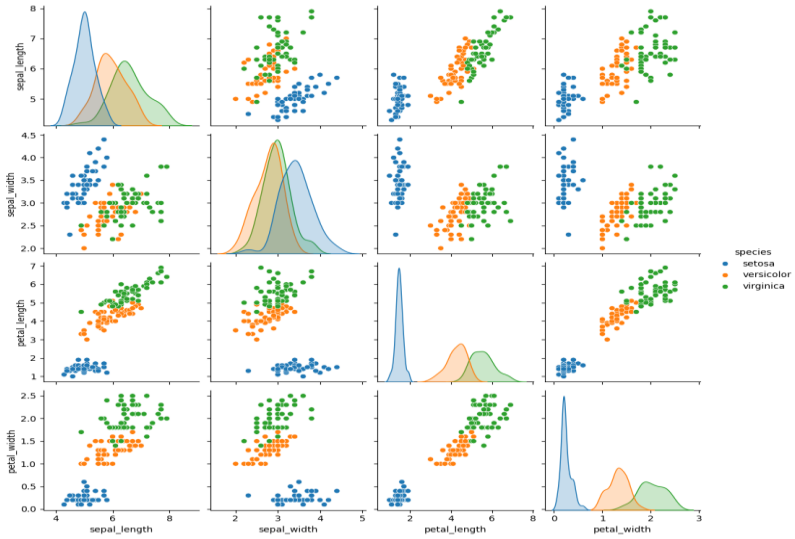
- Histogramas de las características del Iris

Representación de la distribución de las variables numéricas mediante histogramas.



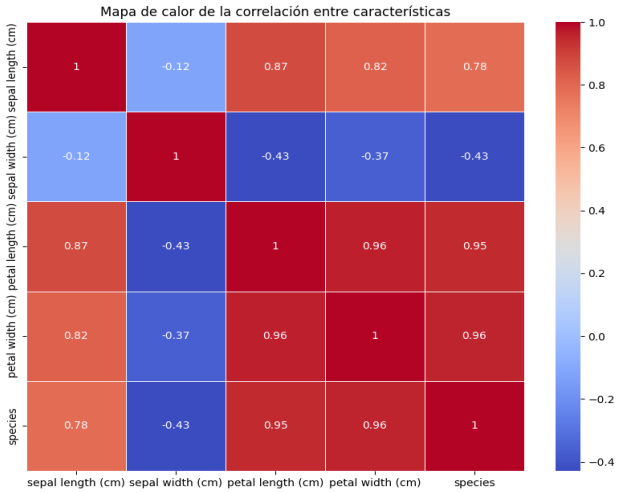
- Pairplot de las características del Iris

Visualización de las relaciones entre las características del conjunto de datos, agrupadas por especie.



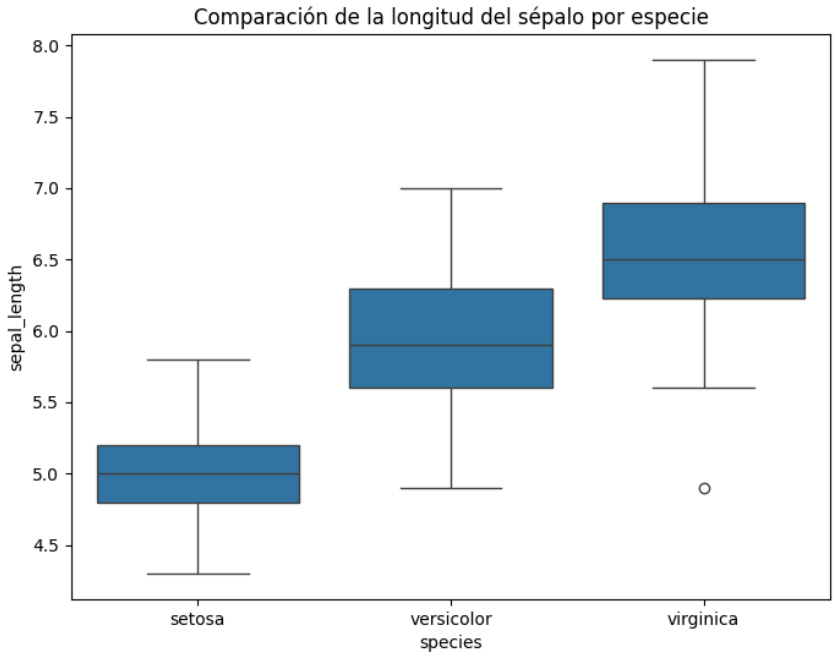
- Mapa de calor de la correlación entre características

Matriz de correlación entre las variables del conjunto de datos para analizar relaciones.



- Comparación de la longitud del sépalo por especie

Boxplot comparando la longitud del sépalo entre las tres especies de Iris.



Actividad 2: Implementación de Modelos de Machine Learning

1. Introducción

En esta actividad, se implementan diversos modelos de Machine Learning para la clasificación y regresión, utilizando el conjunto de datos Iris. Se exploraron modelos como la Regresión Logística, Árboles de Decisión, SVM, KNN, Redes Neuronales, Random Forest y Regresión Lineal.

2. Objetivos

- Implementar diferentes modelos de Machine Learning.
- Evaluar su desempeño en la clasificación de especies de flores.
- Realizar una regresión para predecir una variable continua del conjunto de datos.

3. Desarrollo

3.1. Carga de Datos y Preprocesamiento

Se utilizó la librería seaborn para cargar el conjunto de datos Iris. Posteriormente, se seleccionaron las variables independientes (sepal_length, sepal_width, petal_width) y la variable objetivo (petal_length). Además, se dividió el conjunto de datos en entrenamiento y prueba usando train_test_split.

3.2. Implementación del Modelo de Regresión Lineal

Se utilizó LinearRegression de scikit-learn para entrenar un modelo predictivo de petal_length. Por otra parte se calcularon los coeficientes y la intersección del modelo, realizando predicciones sobre el conjunto de prueba.

Se evaluó el desempeño mediante el error cuadrático medio (MSE) y el coeficiente de determinación (R^2) además de generar una gráfica comparativa entre los valores reales y las predicciones.

3.3. Evaluación de Resultados

Los valores obtenidos de MSE y R^2 indicaron el nivel de ajuste del modelo. Además, la gráfica de dispersión mostró la relación entre valores reales y predichos.

4. Conclusiones

- Se implementó exitosamente un modelo de regresión lineal para predecir petal_length.
- Se observó la importancia de seleccionar adecuadamente las variables predictoras.
- Se identificó la precisión del modelo y su aplicabilidad en problemas de predicción continua.

5. Anexos

Código utilizado para la implementación.

Actividad2 > ActividadDos.py > ...

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC

# Cargar dataset de seaborn (usar un solo dataset)
data = sns.load_dataset('iris')

# Seleccionamos las caracteristicas y variable objetivo
x = data[['sepal_length', 'sepal_width', 'petal_width']]  # Variables independientes
y = data['petal_length']  # Variable objetivo

# Dividir en conjunto de entrenamiento y prueba
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

# Verificar tamaños de los conjuntos
print(f"Tamaño del conjunto de entrenamiento: {x_train.shape}, {y_train.shape}")
print(f"Tamaño del conjunto de prueba: {x_test.shape}, {y_test.shape}")

# Crear y entrenar el modelo de regresión lineal
model = LinearRegression()
model.fit(x_train, y_train)

# Ver coeficientes e intercepto
print(f"Coeficientes: {model.coef_}")
print(f"Intersección (intercepto): {model.intercept_}")

# Realizar predicciones
y_pred = model.predict(x_test)

# Mostrar predicciones
predictions_df = pd.DataFrame({'Real': y_test.values, 'Predicción': y_pred})
print(predictions_df.head())

# Calcular MSE y R²
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Error cuadrático medio (MSE): {mse}")
print(f"Coeficiente de determinación R²: {r2}")

# Graficar resultados
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='blue', label='Predicciones')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', linewidth=2, label="Línea de referencia")
plt.xlabel("Valores reales")
plt.ylabel("Predicciones")
plt.title("Valores reales vs Predicciones")
plt.legend()
plt.show()
```

Gráfica generada