Módulo 3: Análisis de datos con Python (3 horas)

Estadística básica con Python (medias, medianas, desviaciones estándar)

Python, junto con NumPy y Pandas, proporciona herramientas para calcular estadísticas básicas como medias, medianas y desviaciones estándar.

```python
import numpy as np
import pandas as pd

# Ejemplo con NumPy
data = np.array([4, 7, 9, 15, 22])

media = np.mean(data)
mediana = np.median(data)
desviacion_estandar = np.std(data)

print("Media:", media)
print("Mediana:", mediana)
print("Desviación estándar:", desviacion_estandar)

# Ejemplo con Pandas
data = {'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]}
df = pd.DataFrame(data)

media_df = df.mean()
mediana_df = df.median()
desviacion_estandar_df = df.std()

print("\nMedia DataFrame:\n", media_df)
print("\nMediana DataFrame:\n", mediana_df)
print("\nDesviación estándar DataFrame:\n", desviacion_estandar_df)
```

Análisis exploratorio de datos (EDA) con Python

El análisis exploratorio de datos (EDA) es un enfoque para analizar conjuntos de datos con el fin de resumir sus principales características, generalmente utilizando gráficos y estadísticas. Pandas y Seaborn proporcionan herramientas para realizar EDA de manera eficiente.

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Cargar el conjunto de datos de ejemplo de Seaborn
df = sns.load_dataset('iris')
```

```
7
8  # Visualización rápida de las primeras filas del conjunto de datos
9  print(df.head())
10
11 # Resumen estadístico del conjunto de datos
12 print(df.describe())
13
14 # Visualizar la matriz de correlación
15 correlacion = df.corr()
16 sns.heatmap(correlacion, annot=True)
17 plt.show()
18
19 # Visualizar la distribución de una variable
20 sns.displot(df['sepal_length'])
21 plt.show()
22
23 # Visualizar la relación entre dos variables
24 sns.scatterplot(x='sepal_length', y='sepal_width', hue='species', data=df)
25 plt.show()
26
27 # Visualizar la relación entre todas las variables
28 sns.pairplot(df, hue='species')
29 plt.show()
30
31 # Visualizar la distribución de una variable por categoría
32 sns.boxplot(x='species', y='sepal_length', data=df)
33 plt.show()
```

Introducción a la regresión lineal y su implementación con Python

La regresión lineal es una técnica estadística utilizada para modelar la relación entre una variable dependiente (Y) y una o más variables independientes (X). Scikit-learn es una biblioteca de Python que proporciona una implementación fácil de usar para la regresión lineal.

```
1  import numpy as np
2  import pandas as pd
3  import matplotlib.pyplot as plt
4  from sklearn.model_selection import train_test_split
5  from sklearn.linear_model import LinearRegression
6  from sklearn.metrics import mean_squared_error
7
8  # Generar datos de ejemplo
9  X = np.random.rand(100, 1)
10 Y = 2 + 3 * X + np.random.randn(100, 1)
11
12 # Dividir los datos en conjuntos de entrenamiento y prueba
13 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
   random_state=42)
14
```

```
# Crear y entrenar el modelo de regresión lineal
modelo = LinearRegression()
modelo.fit(X_train, Y_train)

# Realizar predicciones en el conjunto de prueba
Y_pred = modelo.predict(X_test)

# Calcular el error cuadrático medio
mse = mean_squared_error(Y_test, Y_pred)
print("Error cuadrático medio:", mse)

# Visualizar la recta de regresión
plt.scatter(X_train, Y_train, color='blue', label='Datos de entrenamiento')
plt.scatter(X_test, Y_test, color='green', label='Datos de prueba')
plt.plot(X_test, Y_pred, color='red', linewidth=2, label='Recta de regresión')
plt.xlabel('Variable independiente X')
plt.ylabel('Variable dependiente Y')
plt.legend()
plt.show()
```

Este ejemplo demuestra cómo crear y entrenar un modelo de regresión lineal utilizando Scikit-learn y evaluar su rendimiento utilizando el error cuadrático medio. Además, muestra cómo visualizar la recta de regresión junto con los datos de entrenamiento y prueba.

Trabajo con datos categóricos

En la práctica, a menudo encontramos variables categóricas en nuestros conjuntos de datos. Para utilizar estas variables en modelos de regresión, necesitamos convertirlas en variables numéricas. Un enfoque común para esto es utilizar la codificación one-hot.

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

# Ejemplo de datos categóricos
data = {'frutas': ['manzana', 'naranja', 'manzana', 'naranja', 'manzana']}
df = pd.DataFrame(data)

# Utilizando pandas para codificación one-hot
df_encoded = pd.get_dummies(df)
print(df_encoded)

# Utilizando scikit-learn para codificación one-hot
encoder = OneHotEncoder(sparse=False)
encoded_array = encoder.fit_transform(df[['frutas']])
column_names = encoder.get_feature_names(['frutas'])
df_encoded_sklearn = pd.DataFrame(encoded_array, columns=column_names)
print(df_encoded_sklearn)
```

Regresión lineal múltiple

La regresión lineal múltiple es una extensión de la regresión lineal que utiliza múltiples variables independientes para predecir una variable dependiente. El proceso de entrenamiento y evaluación es similar al caso de una sola variable independiente.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Cargar un conjunto de datos con múltiples variables independientes
data = pd.read_csv('https://raw.githubusercontent.com/ageron/handson-ml2/master/datasets/housing/housing.csv')
data = data.dropna()  # Eliminar filas con valores faltantes

# Seleccionar variables independientes y dependientes
X = data.drop('median_house_value', axis=1)
Y = data['median_house_value']

# Convertir variables categóricas a numéricas
X = pd.get_dummies(X)

# Dividir 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)

# Crear y entrenar el modelo de regresión lineal múltiple
modelo = LinearRegression()
modelo.fit(X_train, Y_train)

# Realizar predicciones en el conjunto de prueba
Y_pred = modelo.predict(X_test)

# Calcular el error cuadrático medio
mse = mean_squared_error(Y_test, Y_pred)
print("Error cuadrático medio:", mse)
```

Al utilizar regresión lineal múltiple, podemos modelar relaciones más complejas entre las variables. Sin embargo, también debemos tener cuidado con problemas como la multicolinealidad y la maldición de la dimensionalidad. En módulos futuros, exploraremos otras técnicas de aprendizaje automático que pueden manejar relaciones no lineales y de alta dimensión entre las variables.