

Guía de Laboratorio – Construcción y utilización de métodos bayesianos

Sistema de conocimiento:

- Fundamentos de Probabilidad Bayesiana.
- Clasificadores Bayesianos Ingenuos (Naïve Bayes).
- Evaluación de Modelos Probabilísticos.

Objetivo:

- Aplicar el clasificador Naïve Bayes a conjuntos de datos preprocesados.

Bibliografía:

- Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. The MIT Press.
- Johnston, B., & Mathur, I. (2019). Applied supervised learning with Python: Use scikit-learn to build predictive models from real-world datasets and prepare yourself for the future of machine learning. Packt Publishing Ltd.
- Johnston, B., & Mathur, I. (2019). *Applied supervised learning with Python: Use scikit-learn to build predictive models from real-world datasets and prepare yourself for the future of machine learning*. Packt Publishing Ltd.

Introducción

En laboratorios anteriores, hemos explorado modelos de aprendizaje supervisado como K-Vecinos Más Cercanos (KNN) y Árboles de Decisión, que realizan predicciones basadas en la similitud de instancias o en la división recursiva del espacio de características.

En esta sesión, nos adentraremos en un paradigma diferente: el **aprendizaje basado en modelos probabilísticos**. Estos modelos utilizan el **Teorema de Bayes** para calcular la probabilidad de que una instancia pertenezca a una clase determinada, dados sus *features* o características. El clasificador más común dentro de este grupo es el **Naïve Bayes (Bayes Ingenuo)**, notable por su simplicidad, velocidad y sorprendente eficacia, especialmente en problemas de clasificación de texto y diagnóstico.

Fundamentos Teóricos: El Teorema de Bayes

El Teorema de Bayes nos permite actualizar la probabilidad de una hipótesis (por ejemplo, "el paciente tiene cáncer") a la luz de nuevas evidencias (por ejemplo, "el resultado de la prueba es positivo").

Los modelos bayesianos emplean el Teorema de Bayes para calcular la probabilidad posterior de que una instancia pertenezca a una clase dada sus características observadas.

$$P(C_k | X) = \frac{P(X | C_k) \cdot P(C_k)}{P(X)}$$

- $P(C_k | X)$: Probabilidad posterior de la clase C_k dado los datos X .
- $P(X | C_k)$: Verosimilitud o probabilidad de los datos bajo la clase C_k .
- $P(C_k)$: Probabilidad a priori de la clase C_k .
- $P(X)$: Evidencia, probabilidad total de los datos.

Se entrenará el modelo calculando las probabilidades a priori y las condicionales para cada clase.

Clasificación con Naïve Bayes en Scikit-learn

Scikit-learn ofrece varias variantes del clasificador Naïve Bayes, optimizadas para diferentes tipos de datos:

- GaussianNB: Asume que los atributos continuos siguen una distribución normal (Gaussiana).
- MultinomialNB: Ideal para datos de conteo (e.g., conteo de palabras en texto).
- BernoulliNB: Diseñado para atributos binarios/booleanos (e.g., presencia o ausencia de una palabra).

Ejemplo Práctico: Diagnóstico Médico

Utilizaremos el conjunto de datos `breast-cancer.csv`, cuyo objetivo es clasificar tumores como malignos (`M`) o benignos (`B`) basándose en características físicas como el radio, la textura y el perímetro.

Paso 1: Importar bibliotecas y cargar datos

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
import matplotlib.pyplot as plt

# Cargar el dataset
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
# Explorar las primeras filas, verificar valores nulos y el tipo de datos
print(df.head())
print(df.info())
print("Clases:", data.target_names)
```

Paso 2: Preprocesamiento de datos

Necesitamos codificar la variable objetivo (de 'M'/'B' a 1/0) y separar las características (X) de la etiqueta (y).

```
# Codificar la variable objetivo: Maligno = 1, Benigno = 0
print("Distribución original de data.target (0: benigno, 1: maligno):")
print(pd.Series(data.target).value_counts())

# Crear columna objetivo (diagnóstico)
df["diagnosis"] = data.target
print("\nDistribución de df['diagnosis'] después de la asignación inicial (data.target):")
print(df["diagnosis"].value_counts())

# Separar características (X) y variable objetivo (y)
X = df.drop("diagnosis", axis=1)
y = df["diagnosis"]

# Dividir el dataset en conjunto de entrenamiento (70%) y prueba (30%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)
```

Paso 3: Entrenar el modelo GaussianNB

```
# 1. Crear el modelo
model = GaussianNB()

# 2. Entrenar el modelo (Ajustar)
model.fit(X_train, y_train)

# 3. Hacer predicciones sobre el conjunto de prueba
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test) # Obtenemos las probabilidades
```

Paso 4: Evaluar el modelo

Evaluamos la precisión y visualizamos la matriz de confusión.

```
# Precisión general
accuracy = metrics.accuracy_score(y_test, y_pred)
print(f'Precisión (Accuracy) del modelo: {accuracy:.4f}')
```



```
# Matriz de Confusión
# Especificando 'labels=[0, 1]', se asegura que la matriz de confusión es 2x2,
# incluso si falta una clase en y_test o en y_pred.
conf_matrix = metrics.confusion_matrix(y_test, y_pred, labels=[0, 1])
cm_display = metrics.ConfusionMatrixDisplay(conf_matrix, display_labels=['benign', 'malignant'])

cm_display.plot(cmap='Blues')
plt.title('Matriz de Confusión - Clasificador Naïve Bayes')
plt.show()
```



```
# Reporte de Clasificación
# De igual manera, especifica las etiquetas para el informe de clasificación
# para asegurarte de que incluya ambas clases.
print(metrics.classification_report(y_test, y_pred, target_names=['benign', 'malignant'], labels=[0, 1]))
```

Paso 5: Interpretación Probabilística

Una ventaja clave de los modelos bayesianos es la interpretabilidad de las probabilidades.

```
# Veamos las probabilidades predichas para las primeras 5 muestras de prueba.
# La columna 0 es P(Benigno | X), la columna 1 es P(Maligno | X).
print("Probabilidades predichas para las primeras 5 muestras:")
```

```
print(y_pred_proba[:5])  
print("\nClases predichas para las primeras 5 muestras:")  
print(y_pred[:5])
```

Ejercicios

Cree un Jupyter Notebook y resuelva los siguientes ejercicios.

1. Utilice el conjunto de datos wine-quality.csv o cualquier otro dataset de clasificación para entrenar un clasificador Naive Bayes y evalúe su desempeño.
2. Compare el rendimiento del clasificador Naive Bayes con el de un clasificador KNN o Árbol de Decisión en el mismo conjunto de datos.
3. Investigue y explique cómo modificar el modelo para variables categóricas usando Naive Bayes Multinomial y aplíquelo en un conjunto de datos de texto (por ejemplo, clasificación de spam).