Alumno:

Oscar Ivan Valenzuela Diaz

Doctorado:

Sistemas Computaciones

Materia:

Seminario de Programación de Computadoras

Actividad de Aprendizaje:

Actividad 3

Docente de la materia:

Dr. Gandhi Samuel Hernández Chan

Desarrollo

Introducción

El aprendizaje automático se divide en dos categorías principales: aprendizaje supervisado y aprendizaje no supervisado. Ambas categorías tienen sus propias técnicas y algoritmos que se utilizan para resolver problemas específicos de clasificación y agrupamiento (clustering). En esta investigación, exploraremos las características, aplicaciones y diferencias de estos tipos de aprendizaje y proporcionaremos una matriz de comparación con los modelos más utilizados, como KNN, SVM, Regresión Lineal y No Lineal, y Árboles de Decisión.

Aprendizaje Supervisado

El aprendizaje supervisado se basa en el uso de datos etiquetados. Aquí, el modelo se entrena con un conjunto de datos que incluye tanto las características de entrada como las salidas esperadas. Los algoritmos de aprendizaje supervisado aprenden a mapear entradas a salidas basándose en los ejemplos proporcionados durante el entrenamiento.

Ejemplos de Algoritmos Supervisados:

K-Nearest Neighbors (KNN):

Descripción: Algoritmo basado en la proximidad de datos.

Ventajas: Sencillo de implementar, no requiere suposiciones sobre la distribución de los datos.

Desventajas: Sensible a la escala de los datos y al ruido.

Aplicaciones: Reconocimiento de patrones, clasificación de texto.

Support Vector Machines (SVM):

Descripción: Busca el hiperplano que mejor separa las clases en los datos.

Ventajas: Efectivo en espacios de alta dimensión, robusto contra el sobreajuste.

Desventajas: Costoso en términos computacionales, no es adecuado para conjuntos de datos muy grandes.

Aplicaciones: Clasificación de imágenes, detección de spam.

Regresión Lineal y No Lineal:

Descripción: Modela la relación entre variables independientes y dependientes.

Ventajas: Fácil de interpretar, útil para relaciones lineales.

Desventajas: Puede no capturar relaciones complejas, susceptible a los outliers.

Aplicaciones: Predicción de precios, análisis de tendencias.

Árboles de Decisión:

Descripción: Modelo basado en reglas de decisión derivadas de las características de los datos.

Ventajas: Fácil de interpretar, maneja bien los datos categóricos.

Desventajas: Propenso al sobreajuste, puede ser inestable con pequeñas variaciones en los datos.

Aplicaciones: Diagnóstico médico, análisis de riesgos.

Aprendizaje No Supervisado

El aprendizaje no supervisado se utiliza cuando los datos no están etiquetados. Aquí, el modelo intenta encontrar patrones y estructuras ocultas en los datos sin guías externas.

Ejemplos de Algoritmos No Supervisados:

K-Means Clustering:

Descripción: Agrupa los datos en k clústeres basándose en la similitud.

Ventajas: Simple y eficiente, fácil de interpretar.

Desventajas: Necesita definir el número de clústeres k previamente, sensible a la escala de los datos.

Aplicaciones: Segmentación de clientes, análisis de comportamiento.

Hierarchical Clustering:

Descripción: Crea una jerarquía de clústeres mediante un enfoque de división o agrupación.

Ventajas: No necesita especificar el número de clústeres previamente, proporciona una visualización clara (dendrograma).

Desventajas: No escalable para grandes conjuntos de datos, puede ser sensible al ruido y a los outliers.

Aplicaciones: Análisis de genes, agrupamiento de documentos.

PCA (Análisis de Componentes Principales):

Descripción: Reducción de dimensionalidad manteniendo la mayor variabilidad posible.

Ventajas: Reduce la dimensionalidad de los datos, elimina la multicolinealidad.

Desventajas: Puede perder información significativa, interpretabilidad de los componentes.

Aplicaciones: Compresión de datos, visualización de datos.

Matriz de Comparación

Algoritmo	Tipo	Descripción	Ventajas	Desventajas	Aplicaciones
KNN	Supervisado	Basado en la proximidad de los datos	Simple, no requiere suposiciones	Sensible a la escala y al ruido	Reconocimiento de patrones
SVM	Supervisado	Encuentra el mejor hiperplano separador	Efectivo en alta dimensión, robusto contra el sobreajuste	Costoso computacionalmente	Clasificación de imágenes
Regresión Lineal	Supervisado	Modela la relación lineal entre variables	Fácil de interpretar	No captura relaciones complejas, sensible a outliers	Predicción de precios
Árboles de Decisión	Supervisado	Basado en reglas de decisión derivadas de los datos	Fácil de interpretar, maneja datos categóricos	Propenso al sobreajuste, inestabilidad con variaciones	Diagnóstico médico
K-Means	No Supervisado	Agrupa datos en k clústeres basándose en la similitud	Simple y eficiente	Necesita definir k, sensible a la escala	Segmentación de clientes
Hierarchical Clustering	No Supervisado	Crea jerarquía de clústeres mediante enfoque de división	No necesita especificar k, visualización clara	No escalable, sensible a ruido y outliers	Análisis de genes
PCA	No Supervisado	Reducción de dimensionalidad	Reduce dimensionalidad, elimina multicolinealidad	Puede perder información significativa, interpretación	Compresión de datos

Conclusiones

Actividad 2

El análisis de los tipos de aprendizaje automático supervisado y no supervisado y sus respectivos algoritmos de clasificación y agrupamiento nos permite entender mejor las fortalezas y limitaciones de cada método. Esta comprensión es crucial para elegir el algoritmo adecuado según la naturaleza del problema y las características del conjunto de datos. En la siguiente sección, implementamos un programa en Python para clasificar o agrupar los datos obtenidos previamente, evaluando el rendimiento de varios modelos para determinar cuál se ajusta mejor a los datos

Python

```python
# Importar las bibliotecas necesarias
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np

# Cargar el DataFrame desde el archivo CSV generado anteriormente
df = pd.read_csv('tendencias_elimparcial.csv')

# Eliminar filas con valores NaN en la columna 'noticias'
df = df.dropna(subset=['noticias'])

# Convertir textos de noticias en vectores numéricos
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(df['noticias'])
y = df['categoria']

# 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)

# Definir una lista de modelos a probar
modelos = {
    'Regresión Logística': LogisticRegression(),
    'Random Forest': RandomForestClassifier(),
    'SVM': SVC()
}

# Entrenar y evaluar cada modelo
resultados = {'Modelo': [], 'Precisión': []}
mejor_modelo = None
mejor_precision = 0

for nombre, modelo in modelos.items():
    modelo.fit(X_train, y_train)
    predicciones = modelo.predict(X_test)
    precision = accuracy_score(y_test, predicciones)
    resultados['Modelo'].append(nombre)
    resultados['Precisión'].append(precision)
    print(f"{nombre}: {precision}")
    if precision > mejor_precision:
        mejor_precision = precision
        mejor_modelo = nombre

print(f"El mejor modelo es: {mejor_modelo} con una precisión de
{mejor_precision}")
```

```python
# Mostrar los resultados como un DataFrame
resultados_df = pd.DataFrame(resultados)
print("\nTabla de Comparación:")
print(resultados_df)
```

```
vaos@ATUSLNXVAOS01:~/Documents/python$ /bin/python3 /home/vaos/Documents/python/modelos.py
/home/vaos/.local/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Regresión Logística: 0.7142857142857143
Random Forest: 0.5714285714285714
SVM: 0.5714285714285714
El mejor modelo es: Regresión Logística con una precisión de 0.7142857142857143

Tabla de Comparación:
               Modelo  Precisión
0  Regresión Logística   0.714286
1        Random Forest   0.571429
2                  SVM   0.571429
```

Explicación:

Importación de bibliotecas: Se importan las bibliotecas necesarias para el procesamiento de datos y el aprendizaje automático, incluyendo Scikit-learn (para los modelos y las métricas), Pandas (para manipulación de datos) y NumPy (para operaciones numéricas).

Carga de datos: Se carga el DataFrame desde el archivo CSV generado anteriormente (tendencias_elimparcial.csv) utilizando la función pd.read_csv() de Pandas.

Limpieza de datos: Se eliminan las filas que contienen valores NaN en la columna 'noticias' utilizando el método dropna() de Pandas. Esto se hace para asegurarse de que los datos estén limpios y no haya valores faltantes que puedan causar problemas durante el entrenamiento del modelo.

Vectorización de texto: Se utiliza CountVectorizer() de Scikit-learn para convertir los textos de las noticias en vectores numéricos. Esto es necesario para que los algoritmos de aprendizaje automático puedan trabajar con los datos de texto.

División de datos: Se dividen los datos en conjuntos de entrenamiento y prueba utilizando train_test_split() de Scikit-learn. Esto se hace para evaluar el rendimiento del modelo en datos que no ha visto durante el entrenamiento.

Definición de modelos: Se crea una lista de modelos a probar, que incluye Regresión Logística, Random Forest y SVM (Support Vector Machine).

Entrenamiento y evaluación de modelos: Se itera sobre cada modelo en la lista y se entrena con los datos de entrenamiento (X_train y y_train). Luego, se realizan predicciones en los datos de prueba (X_test) y se evalúa la precisión del modelo utilizando accuracy_score() de Scikit-learn.

Selección del mejor modelo: Se comparan las precisiones de cada modelo y se selecciona el modelo con la precisión más alta como el mejor modelo.

Visualización de resultados: Se muestran los resultados en forma de tabla utilizando un DataFrame de Pandas, que incluye el nombre de cada modelo y su precisión.

Con esta actividad tratamos de proporciona una forma estructurada de comparar varios modelos de aprendizaje automático y determinar cuál tiene el mejor rendimiento en la clasificación de noticias según sus categorías.

Referencias:

- Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

- Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.

- Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd ed.). O'Reilly Media.

- Raschka, S., & Mirjalili, V. (2019). Python Machine Learning (3rd ed.). Packt Publishing.

- Jurafsky, D., & Martin, J. H. (2008). Speech and Language Processing (2nd ed.). Prentice Hall.

- Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media.

- Sarkar, D. (2019). Text Analytics with Python: A Practitioner's Guide to Natural Language Processing. Apress.

- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning: With Applications in R. Springer.

- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer.

- Provost, F., & Fawcett, T. (2013). Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking. O'Reilly Media.