1 DE MAYO DE 2024

EJERCICIO 2 Y 3 - CLASIFICADORES

PRACTICA 2

CRUZ GALLEGOS RAMSES AARÓN
MORALES ZEPEDA IVAN YUTLANIH
SEMINARIO DE SOL DE PROBLEMAS DE INTELIGENCIA ARTIFICIAL 2
D05
MTRO: CAMPOS PEÑA DIEGO

Índice

Introducción

Los clasificadores como Regresión Logística (Logistic Regression), K-Vecinos Cercanos (K-Nearest Neighbors), Máquinas de Vectores de Soporte (Support Vector Machines) y Naive Bayes son muy importantes al asignar categorías a diferentes instancias de datos. Cada uno de estos métodos tiene sus propias características, ventajas y desventajas que los hacen adecuados para diferentes tipos de problemas de clasificación. Se analizará su desempeño utilizando tres conjuntos de datos diferentes: Swedish Auto Insurance, Wine Quality y Pima Indians Diabetes.

Desarrollo

Código

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
import warnings

# Desactivar las advertencias de Sklearn
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=RuntimeWarning)

# Cargar los datasets desde los archivos CSV
swedish_auto_data = pd.read_csv('AutoInsurSweden.csv')
wine_quality_data = pd.read_csv('wine-Quality.csv')
pima_diabetes_data = pd.read_csv('pima-indians-diabetes.csv')

# Categorizar los valores de Y en Swedish Auto Insurance Dataset
quartiles = swedish_auto_data['Y'].quantile([0.25, 0.5, 0.75])
low_limit = quartiles.iloc[0]
medium_limit = quartiles.iloc[1]
high_limit = quartiles.iloc[2]

def categorize_y(y):
    if y <= low_limit:
        return 'bajo'
    elif y <= medium_limit:
        return 'medio'
    else:
```

```
        return 'alto'

swedish_auto_data['Y_category'] =
swedish_auto_data['Y'].apply(categorize_y)

# Definir una función para implementar clasificadores y calcular
métricas de evaluación
def evaluate_classifier(X, y, classifier):
    # Dividir los datos en conjunto de entrenamiento y conjunto de
prueba
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

    # Entrenar el clasificador
    if classifier == LogisticRegression:
        model = LogisticRegression(max_iter=1000)
    else:
        model = classifier()
    model.fit(X_train, y_train)

    # Hacer predicciones
    y_pred = model.predict(X_test)

    # Calcular métricas de evaluación
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted',
zero_division='warn')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')

    # Imprimir métricas
    print("\nMétricas: ", classifier.__name__)
    print("Accuracy:", accuracy)
    print("Precision:", precision)
    print("Recall:", recall)
    print("F1 Score:", f1)

# Swedish Auto Insurance Dataset
print("\nSwedish Auto Insurance Dataset:")
X_swedish = swedish_auto_data[['X']]
y_swedish = swedish_auto_data['Y_category']
evaluate_classifier(X_swedish, y_swedish, LogisticRegression)
evaluate_classifier(X_swedish, y_swedish, KNeighborsClassifier)
evaluate_classifier(X_swedish, y_swedish, SVC)
evaluate_classifier(X_swedish, y_swedish, GaussianNB)
evaluate_classifier(X_swedish, y_swedish, MLPClassifier)

# Wine Quality Dataset
```

```
print("\nWine Quality Dataset:")
X_wine = wine_quality_data.drop('quality', axis=1)
y_wine = wine_quality_data['quality']
evaluate_classifier(X_wine, y_wine, LogisticRegression)
evaluate_classifier(X_wine, y_wine, KNeighborsClassifier)
evaluate_classifier(X_wine, y_wine, SVC)
evaluate_classifier(X_wine, y_wine, GaussianNB)
evaluate_classifier(X_wine, y_wine, MLPClassifier)

# Pima Indians Diabetes Dataset
print("\nPima Indians Diabetes Dataset:")
X_pima = pima_diabetes_data.drop('Class variable (0 or 1)', axis=1)
y_pima = pima_diabetes_data['Class variable (0 or 1)']
evaluate_classifier(X_pima, y_pima, LogisticRegression)
evaluate_classifier(X_pima, y_pima, KNeighborsClassifier)
evaluate_classifier(X_pima, y_pima, SVC)
evaluate_classifier(X_pima, y_pima, GaussianNB)
evaluate_classifier(X_pima, y_pima, MLPClassifier)
```

El código implementado utiliza las bibliotecas sklearn y pandas en Python para cargar los conjuntos de datos, categorizar los valores de destino en el conjunto de datos Swedish Auto Insurance en tres categorías (bajo, medio, alto), y evaluar el rendimiento de cinco clasificadores diferentes en cada conjunto de datos.

Se definió la función: **evaluate_classifier**, la cual toma como entrada un conjunto de características (X) y un vector de etiquetas (y), junto con un clasificador específico, divide los datos en conjuntos de entrenamiento y prueba **(80% y 20%)**, entrena el clasificador en los datos de entrenamiento y luego evalúa su rendimiento en los datos de prueba. Calcula y muestras métricas de evaluación como precisión, exactitud, recall y F1-score.

Las siguientes métricas se calculan para cada clasificador:

- Accuracy: La proporción de predicciones correctas sobre el total de predicciones.
- Precision: La proporción de verdaderos positivos sobre el total de predicciones positivas.
- Recall: La proporción de verdaderos positivos sobre el total de valores verdaderos en el conjunto de datos.
- F1-score: La media armónica de precisión y recall, que proporciona una medida de precisión equilibrada.

Análisis de Resultados

En general, el mejor clasificador varía según el conjunto de datos y la tarea específica. Es importante considerar las métricas de evaluación y las características de cada conjunto de datos para seleccionar el clasificador más adecuado.

Comparación del rendimiento de los algoritmos

Dataset	Algoritmo	Accuracy	Precision	Recall	F1-Score
Swedish Auto Insurance	Logistic Regression	0.6923	0.8654	0.6923	0.7433
	KNeighborsClassifier	0.6154	0.8615	0.6154	0.6838
	SVC	0.6923	0.8718	0.6923	0.7510
	Naive Bayes	0.5385	0.8718	0.5385	0.6357
	MLPClassifier	**0.8462**	0.7160	**0.8462**	**0.7756**
Wine Quality	Logistic Regression	0.5750	0.5287	0.5750	0.5405

	KNeighborsClassifier	0.4563	0.4223	0.4563	0.4299
	SVC	0.5094	0.5645	0.5094	0.4618
	Naive Bayes	0.5500	0.5423	0.5500	0.5455
	MLPClassifier	0.5656	0.5435	0.5656	0.5268
Pima Indians Diabetes	Logistic Regression	0.7468	0.7502	0.7468	0.7482
	KNeighborsClassifier	0.6623	0.6712	0.6623	0.6658
	SVC	**0.7662**	0.7613	**0.7662**	0.7586
	Naive Bayes	**0.7662**	0.7707	**0.7662**	0.7679
	MLPClassifier	0.7078	0.6967	0.7078	0.6901

Mejor método para cada dataset:

Dataset	Best Classifier (Based on F1-Score)	Accuracy	Precision	Recall
Swedish Auto Insurance	SVC	0.6923	0.8718	0.6923
Wine Quality	SVC	0.5094	0.5645	0.5094
Pima Indians Diabetes	SVC and Naive Bayes (tied)	0.7662	0.7613 and 0.7707	0.7662 and 0.7662

En resumen, la interpretación general de los resultados es la siguiente:

- **Swedish Auto Insurance:** SVC y Regresión Logística funcionan bien en general, con un buen equilibrio entre precisión y recuerdo. MLPClassifier logra una alta precisión pero tiene un puntaje F1 más bajo, lo que sugiere un posible sobreajuste.
- **Wine Quality:** SVC se destaca como el mejor desempeño, dado su puntaje F1 relativamente más alto a pesar de la menor precisión. Este conjunto de datos podría ser más desafiante debido a su complejidad inherente.
- **Pima Indians Diabetes:** SVC y Naive Bayes comparten el primer puesto, demostrando su eficacia en la clasificación de pacientes diabéticos. La regresión logística y KNN también exhiben un buen desempeño.

Conclusión

Tras analizar los resultados obtenidos, podemos observar que el desempeño de cada clasificador varía según el conjunto de datos en cuestión. En el caso del conjunto de datos Swedish Auto Insurance, el mejor desempeño se logra con la Red Neuronal (MLPClassifier), mientras que en el conjunto de datos Pima Indians Diabetes, tanto las Máquinas de Vectores de Soporte (SVC) como Naive Bayes (GaussianNB) muestran un rendimiento destacado.

La elección del clasificador más adecuado para un problema específico dependerá de varios factores, como la naturaleza de los datos y los requisitos específicos de precisión, velocidad y escalabilidad.