Reporte del proyecto Clasificadores

Hector Vazquez Terreros

Programación 9292

Un clasificador es un tipo de modelo de aprendizaje automático que se utiliza para categorizar o etiquetar datos en diferentes clases o grupos, basándose en características de entrada. Forma parte del aprendizaje supervisado, donde el algoritmo se entrena con datos etiquetados (datos que incluyen tanto las características de entrada como las etiquetas correctas). El objetivo de un clasificador es aprender la relación entre las características de entrada y las etiquetas de las clases, para poder predecir correctamente la clase de nuevos datos.

En el proyecto se desarrolla un sistema de clasificación utilizando diferentes algoritmos de aprendizaje automático. El objetivo es implementar clasificadores y comparar varios de estos aplicados a un conjunto de datos sobre cáncer para determinar cuál de ellos proporciona los mejores resultados de precisión y mas indicadores de rendimiento.

Se hizo uso de los siguientes clasificadores para realizar la práctica, y con ello el entrenamiento y evaluación:

- Regresión logística.
- k-Nearest Neighbors (k-NN).
- Máquinas de soporte vectorial (SVM).
- Árboles de decisión.
- Random Forest.

Los modelos fueron implementados utilizando la librería sklearn en Python y entrenados con un conjunto de datos etiquetados. Para la evaluación de los modelos, se emplearon métricas como la matriz de confusión, el reporte de clasificación (precisión, recall, F1-score), y la comparación entre las precisiones obtenidas por cada clasificador.

Implementación

1. **Importación de bibliotecas**

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report
```

2. **Clase Modelos_Clasificadores**

```
class Modelos_Clasificadores:

    def __init__(self):
        self.modelos_disponibles = {
            "Logística": LogisticRegression(),
            "Máquina de soporte vectorial": SVC(),
            "k-Nearest Neighbors": KNeighborsClassifier(),
            "Árbol de decicion": DecisionTreeClassifier(),
            "Bosque Aleatorio": RandomForestClassifier()
        }
        self.resultados_modelos = {}

    def ajustar_modelos(self, datos_entrenamiento, etiquetas_entrenamiento):
        for nombre, modelo in self.modelos_disponibles.items():
            modelo.fit(datos_entrenamiento, etiquetas_entrenamiento)
            print("El modelo " + nombre + " ha sido entrenado.")

    def evaluar_modelos(self, datos_prueba, etiquetas_prueba):
        for nombre, modelo in self.modelos_disponibles.items():
            predicciones = modelo.predict(datos_prueba)
            print("\nEvaluación del modelo: " + nombre)
            print("\nMatriz de confusión:")
            print(confusion_matrix(etiquetas_prueba, predicciones))
            print("\nReporte de clasificación:")
            print(classification_report(etiquetas_prueba, predicciones))
            self.resultados_modelos[nombre] =
    classification_report(etiquetas_prueba, predicciones, output_dict=True)
```

3. **Funcion base**

```
def funcion_base(): Rt_ach = '/content/cancer - cancer (1).csv' conj_dat =
pd.read_csv(Rt_ach) etiquetas_numericas = []

    for valor in conj_dat['diagnosis']:
      if valor == 'M':
        etiquetas_numericas.append(1)
      else:
        etiquetas_numericas.append(0)

    conj_dat['diagnosis'] = etiquetas_numericas
    Alpha = conj_dat.drop(['id', 'diagnosis'], axis=1)
    Beta = conj_dat['diagnosis']
    Alpha_entrenamiento, Alpha_prueba, Beta_entrenamiento, Beta_prueba =
    train_test_split(Alpha, Beta, test_size=0.30, random_state=42)

    escalador = StandardScaler()
    Alpha_entrenamiento = escalador.fit_transform(Alpha_entrenamiento)
    Alpha_prueba = escalador.transform(Alpha_prueba)

    modelos = Modelos_Clasificadores()
    print("\nEntrenando modelos...")
    modelos.ajustar_modelos(Alpha_entrenamiento, Beta_entrenamiento)

    print("\nEvaluando modelos...")
    modelos.evaluar_modelos(Alpha_prueba, Beta_prueba)

    precisiones_modelos = {}
    for nombre, resultados in modelos.resultados_modelos.items():
      precisiones_modelos[nombre] = resultados['accuracy']

    print("\nComparación de precisión de los modelos:")
    for nombre, precision in precisiones_modelos.items():
      print(f"{nombre}: {round(precision, 9)}")
```

return precisiones_modelos

4. **Análisis Comparativo**

```
precisiones_modelos = {
   'Logística': 0.993333333,
   'Máquina de soporte vectorial': 0.98,
   'k-Nearest Neighbors': 0.986666667,
   'Árbol de decicion': 0.94,
   'Bosque Aleatorio': 0.966666667
}
```

5. **Grafica de comparacion**

```
def generar_tabla_resultados(precisiones_modelos):

plt.figure(figsize=(15, 5))

plt.bar(precisiones_modelos.keys(), precisiones_modelos.values(), color='#9a70b0')

plt.title("Comparación entre modelos clasificadores")

plt.ylabel("Precisión de cada modelo") plt.xlabel("Modelo utilizado")

plt.xticks(rotation=0)

plt.yticks([i/20 for i in range(0, 21)])

plt.show()
```

Resumen del Proceso

1. Entrenamiento (ajustar_modelos):

   Ajustamos los parámetros de los clasificadores con los datos de entrenamiento. Utilizamos el método .fit() para entrenar los modelos con los datos de entrenamiento (Alpha_entrenamiento y Beta_entrenamiento).

2. Evaluación (evaluar_modelos):
   Evaluamos el rendimiento de los modelos entrenados con datos de prueba. Usamos el método .predict() para realizar predicciones y calcular las métricas utilizando los datos de prueba (Alpha_prueba y Beta_prueba).

3. Resultados organizados:
   Se almacenaron métricas como la precisión y F1-score de cada clasificador en el diccionario self.resultados_modelos.

¿Como se hizo?

Se convirtieron las etiquetas de diagnóstico ('M' y 'B') en valores numéricos (1 y 0 respectivamente) usando un ciclo for.

Además, se excluyó la columna id y la columna diagnosis para que solo las características relevantes sean utilizadas en el modelo.

- Entrenamiento (70%): Se usaron estos datos para entrenar los modelos.
- Prueba (30%): Se reservaron para evaluar el desempeño de los modelos entrenados.
- Normalización: Se utilizó StandardScaler de sklearn para escalar las características, evitando sesgos debido a diferentes rangos de los valores.
- Se entrenaron los cinco clasificadores utilizando el método .fit() de sklearn, que ajusta los parámetros internos del modelo para aprender a clasificar los datos.

Clasificadores entrenados:

1. Regresión Logística: Modelo lineal que ajusta una recta o hiperplano para separar las clases.
2. k-NN: Clasifica los puntos en función de la clase de sus vecinos más cercanos.
3. SVM (Máquinas de Soporte Vectorial): Encuentra el hiperplano óptimo que divide las clases.
4. Árbol de Decisión: Utiliza una serie de reglas para tomar decisiones y clasificar los datos.
5. Random Forest: Conjunto de árboles de decisión que mejora la precisión combinando los resultados de varios árboles.

Resultados obtenidos:

- Todos los modelos fueron entrenados exitosamente.
- Regresión Logística y k-NN mostraron un buen desempeño con precisiones cercanas a 1 durante el entrenamiento.

Evaluación de los clasificadores

Después de entrenar los modelos, utilizamos .predict() para hacer predicciones sobre los datos de prueba. Una vez que el modelo está entrenado con .fit(), .predict() genera predicciones sobre datos nuevos usando los parámetros ajustados durante el entrenamiento.

Métricas calculadas:

- Matriz de confusión: Muestra la comparación entre las predicciones y las etiquetas reales para evaluar el desempeño del modelo.
- Reporte de clasificación: Proporciona métricas clave como precisión, recall, y F1-score para evaluar el rendimiento de los clasificadores.

Resultados obtenidos:

- Regresión Logística y k-NN destacaron con una precisión de 0.99 o cercana.
- Otros clasificadores como Árbol de Decisión y Random Forest alcanzaron precisiones de 0.92 y 0.96, respectivamente.

Comparación y selección del mejor modelo

El modelo con la mayor precisión fue seleccionado como el mejor para esta tarea de clasificación.

Resultados: Regresión Logística fue el mejor modelo, con una precisión de 0.993.

Resumen del Flujo de Trabajo

1. Definición de clasificadores: Se implementaron y entrenaron los clasificadores utilizando el método .fit().
2. Preparación de los datos: Se realizaron transformaciones, escalado y división en conjuntos de entrenamiento y prueba.
3. Entrenamiento: Se ajustaron los parámetros de cada modelo con los datos de entrenamiento.
4. Evaluación: Se generaron predicciones con .predict() y se calcularon las métricas de rendimiento.

5. Comparación: Se identificó el mejor modelo en base a la precisión obtenida durante la evaluación.