INSTITUTO POLITÉCNICO NACIONAL

ESCOM

INTELIGENCIA ARTIFICIAL

LAB 10: KNN & BAYES

MARTÍNEZ CHÁVEZ JORGE ALEXIS

6CV3

22 NOVIEMBRE 2024

Introducción

Los clasificadores son herramientas fundamentales para resolver problemas en los que se requiere predecir o asignar etiquetas a nuevos datos. Entre los clasificadores más utilizados por su simplicidad y eficacia se encuentran Naive Bayes y K-Nearest Neighbors (KNN). A pesar de tener enfoques conceptualmente distintos, ambos han demostrado ser soluciones poderosas para una amplia gama de problemas de clasificación. Naive Bayes es un clasificador probabilístico basado en el teorema de Bayes, el cual calcula la probabilidad de que una instancia pertenezca a una clase específica dado un conjunto de características a diferencia de K-Nearest Neighbors (KNN) se basa en un enfoque completamente diferente: el aprendizaje basado en instancias. KNN no realiza un entrenamiento explícito; en lugar de eso, cuando se presenta una nueva instancia para clasificar, el algoritmo busca los k vecinos más cercanos en el espacio de características y asigna la clase más común entre ellos.

Desarrollo

Para esta practica vamos a separar en dos archivos, el primero para el clasificador de bayes y el segundo para el de knn, cada uno lo haremos con los 3 metodos de validación, y vamos a determinar su acurracy y su matriz de confusión.

Primero para el de Bayes vamos a hacer su validación para hold out, 10 fold cross validation y leave one out

```python
# Ejecutar Naive Bayes
def ejecutar_naive_bayes(X, y):
    modelo = GaussianNB()
    validaciones = {
        "Hold-Out (70/30)": hold_out,
        "10-Fold Cross-Validation": lambda m, X, y: cross_validation(m, X, y, 10),
        "Leave-One-Out": leave_one_out
    }

    resultados = []
    for metodo, funcion_validacion in validaciones.items():
        accuracy, matriz_confusion = funcion_validacion(modelo, X, y)
        resultados.append((metodo, accuracy, matriz_confusion))
    return resultados
```

Primero vamos a hacer el de Hold Out, en donde debemos de dividri 70/30 para entrenar el modelo y probar el modelo respectivamente, una vez hecha la división debemos de hacer el entrenamiento y después hacer la predicción. Por ultimo calculamos el accuracy y la matriz de confusión.

```python
# Métodos de validación
def hold_out(modelo, X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
    modelo.fit(X_train, y_train)
    y_pred = modelo.predict(X_test)
    return accuracy_score(y_test, y_pred), confusion_matrix(y_test, y_pred)
```

Ahora hacemos para cross validation, en donde el conjunto de datos lo vamos a subdividir en 10 subconjuntos, entrenamos en 9 de ellos y el sobrante lo vamos a dejar como prueba, y este proceso lo vamos a repetir 10 veces para que almenos una vez cada subconjunto se el subconjunto de prueba. Se calcula el promedio de las precisiones obtenidas en las 10 iteraciones para obtener una evaluación más estable y generalizable. Finalmente de la misma forma calculamos su accuracy y lal matriz de confusión.

```python
def cross_validation(modelo, X, y, folds):
    kf = KFold(n_splits=folds, shuffle=True, random_state=42)
    scores = cross_val_score(modelo, X, y, cv=kf)
    modelo.fit(X, y)
    y_pred = modelo.predict(X)
    conf_matrix = confusion_matrix(y, y_pred)
    return np.mean(scores), conf_matrix
```

Ahora lo hacemos para el leave one out, en donde se divide en tantas particiones como instancias tenga el dataset, se entrena el modelo con todos los datos menos una única instancia. Y se evalúa el modelo en esa instancia excluida. De la misma forma calculamos su accuracy y lal matriz de confusión.

```python
def leave_one_out(modelo, X, y):
    loo = LeaveOneOut()
    y_true, y_pred = [], []
    for train_idx, test_idx in loo.split(X):
        modelo.fit(X[train_idx], y[train_idx])
        pred = modelo.predict(X[test_idx])
        y_true.extend(y[test_idx])
        y_pred.extend(pred)
    return accuracy_score(y_true, y_pred), confusion_matrix(y_true, y_pred)
```

Una vez que hayamos obtenido esos resultados, debemos de regresarlos para poder ciclarlos y poder obtener sus resultados

```python
  resultados = []
  for metodo, funcion_validacion in validaciones.items():
      accuracy, matriz_confusion = funcion_validacion(modelo, X, y)
      resultados.append((metodo, accuracy, matriz_confusion))
  return resultados
```

Despues unicamente imprimos los resultados

```python
if __name__ == "__main__":
    datos = cargar_dataset()
    if datos is not None:
        X = datos.iloc[:, :-1].values
        y = datos.iloc[:, -1].values
        resultados = ejecutar_naive_bayes(X, y)

        for res in resultados:
            print(f"Método de Validación: {res[0]}")
            print(f"Accuracy: {res[1]:.4f}")
            print("Matriz de Confusión:")
            print(res[2])
    else:
        print("No se pudo procesar el dataset.")
```

En donde obtenemos el los siguientes resultados, en donde para el hold out nos dice que tenemos un accuracy del 97% donde el modelo logro clasificar de manera correcta, en donde en la clase 1 hubo 19 instancias correctamente clasificadas y 0 errores, en la segunda clase hubo 12 instancias y 1 que se clasifico como clase 3 y por ultimo en la clase 3 13 instancias y 0 errores. Para el caso de cross validation debemos de tener un accuracy del 96% en donde en clas clase 1 hubo 50 instancias con 0 errores, en la clase 2 hubo 47 instancias correctas mas 3 errores con clase 3, en la clase 3 s ehicerón 47 instancias con 3 errores con clase 2.

```
Método de Validación: Hold-Out (70/30)
Accuracy: 0.9778
Matriz de Confusión:
[[19  0  0]
 [ 0 12  1]
 [ 0  0 13]]
Método de Validación: 10-Fold Cross-Validation
Accuracy: 0.9600
Matriz de Confusión:
[[50  0  0]
 [ 0 47  3]
 [ 0  3 47]]
Método de Validación: Leave-One-Out
Accuracy: 0.9533
Matriz de Confusión:
[[50  0  0]
 [ 0 47  3]
 [ 0  4 46]]
PS C:\Users\jorge\OneDrive\Documentos\JORGE\ESCOM\9_SEMESTRE>
```

Ahora para el caso de knn usamos los mismos metodos de validación que usamos con bayes, solo que ahora la forma en la que calsificamos mandamos llamar la nueva función.

```python
# Ejecutar KNN
def ejecutar_knn(X, y):
    modelo = KNeighborsClassifier(n_neighbors=5)
    validaciones = {
        "Hold-Out (70/30)": hold_out,
        "10-Fold Cross-Validation": lambda m, X, y: cross_validation(m, X, y, 10),
        "Leave-One-Out": leave_one_out
    }

    resultados = []
    for metodo, funcion_validacion in validaciones.items():
        accuracy, matriz_confusion = funcion_validacion(modelo, X, y)
        resultados.append((metodo, accuracy, matriz_confusion))
    return resultados
```

Si ejecutamos obtenemos los siguientes resultados en donde nos dice que para el Hold out se tuvo un accuracy del 100% en donde en la clase 1 hubo 19 instancias con 0 errores, en la clase 2 hubo 13 instancias con 0 errores y por ultimo en la clse 3 hubo 13 instancias con 0 errores. Para cross validation en la clase 1 hubo 50 instancias con 0 errores, en la clase 2 hubo 47 instancias con 3 errores con la clase 3 y por ultimo para la clase 3 hubo 48 instancias con 2 errores con la clase 2. Para leave one out en la clase 1 hubo 50 instancias con 0 errores, en la clase 2 hubo 47 instancias con 3 errores con la clase 3 y por ultimo para la clase 3 hubo 48 instancias con 2 errores

```
PS C:\Users\jorge\OneDrive\Documentos\JORGE\ESCOM\9_SEMESTRE> & C:/Users/jorge/AppData/Local/Programs/Python/Python310/python.exe c:/Users/jorge/OneDrive/Documentos/JORGE/ESCOM/9_SEMESTRE/IA/LABORATORIO/10_KNN_BAYES/
y
Método de Validación: Hold-Out (70/30)
Accuracy: 1.0000
Matriz de Confusión:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
Método de Validación: 10-Fold Cross-Validation
Accuracy: 0.9733
Matriz de Confusión:
[[50  0  0]
 [ 0 47  3]
 [ 0  2 48]]
Método de Validación: Leave-One-Out
Accuracy: 0.9667
Matriz de Confusión:
[[50  0  0]
 [ 0 47  3]
 [ 0  2 48]]
PS C:\Users\jorge\OneDrive\Documentos\JORGE\ESCOM\9_SEMESTRE> 
```

Para ambos casos la forma de cargar el data set es de la siguiente forma:

```python
# Carga el archivo con la ruta especificada en el código
def cargar_dataset():
    file_path = "C:/Users/jorge/OneDrive/Documentos/JORGE/ESCOM/9_SEMESTRE/IA/LABORATORIO/10_KNN_BAYES/bezdekIris.data"
    try:
        datos = pd.read_csv(file_path, header=None)
        return datos
    except Exception as e:
        print(f"Error al cargar el archivo: {e}")
        return None
```

Enlace

En el siguiente enlace se encuentra la carpeta de la practica 10
https://github.com/Jorge300403/IA_6CV3_MartinezChavez/tree/main/LAB_10_KNN_BAYES