Memoria - Práctica Aprendizaje Automático

Autor: Hugo Salvador Aizpún

Grado: Inteligencia Artificial

Asignatura: Aprendizaje Automático I

Índice de contenidos

1ª Parte - Entrenamiento

1 - Preprocesado sobre los ejemplos

1.1 - Codificar etiquetas a enteros

La estructura de la etiqueta dada es `[Score;Air_Quality]` a si que debemos de codificar la columna de `Aiquality`, para ello hemos usado `LableEncoder`

1.2 - Creación del pipeline de preprocesado

Como hemos visto en clase, todas las tareas de preprocesado las podemos concentrar en una única tubería con las siguientes acciones:

Estandarización

Hacemos una estandarización de los datos antes de las interacciones para que no salgan números demasiado "grandes" que dominen numéricamente al resto

Interacciones (ingeniería de características)

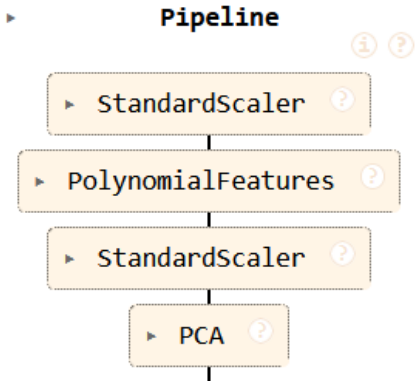
Con objetivo de enriquecer al representación de los datos con interacciones aplicamos `PolynomialFeatures` para que nuestro modelo pueda percibir relaciones conjuntas de dos variables sin diseñar manualmente cada término.

Estandarización

Después de hacer las interacciones puede que la varianza haya aumentado mucho en alguna variable respecto al resto, por ello antes de hacer PCA hacemos una segunda estandarización.

PCA (Análisis de componentes principales)

Reducimos la dimensionalidad del espacio de características quedándonos con el 95% de la varianza total. Es interesante hacer PCA puesto que después de hacer interacciones el número de variables ha aumentado considerablemente.



2 - Construcción de un modelo a priori de la etiqueta

Este es el resultado tras crear nuestro modelo a priori:
`priors:`

```
Modelo a priori / P(y):
 Air_Quality
Good         0.4
Moderate     0.3
Poor         0.2
Hazardous    0.1
```

3 y 4- Construcción de los modelos de verosimilitud y los sistemas clasificadores

Para cada uno de los tres modelos de verosimilitud diseñados, se ha encadenado el preprocesado, el modelo a priori y la verosimilitud en un único objetivo capaz de recibir los datos y devolver etiquetas predichas.

3.1 - Naive Bayes

- Se utiliza `sklearn.naive_bayes.GaussianNB`, pasando explícitamente las `priors` calculadas.
- Al ajustar con `.fit` se estiman medias y varianzas univariantes por clase

3.2 - MVN

- Se aprovecha `sklearn.dicriminant_analysis.QuadraticDiscriminantAnalysis`, que modela cada $p(x|y)$ como gaussiana multivariante con covarianza propia.
- También le proporcionamos las mismas `priors`.

3.3 - GMM

- Se define el modelo con `GMMClassifier` que, en si `fit(x,y)`, entrena un `GaussianMixture` (2 componentes, covarianza completa, EM con varias inicializaciones y regularización ligera) por cada clase.
- En `predict(x)` calcula para cade clase el $log-likelihood$ (`.score_samples`) más el $log-prior$, y elige la clase de mayor puntuación

5 - Comparación de los 3 sistemas

Tenemos que decidir que clasificador implantaremos en nuestro sistema. Por ello hemos seguido los siguientes pasos:

- Protocolo de validación
  - Nos quedamos con un subconjunto de validación usando `stratify = y_enc` para mantener la proporción entre clases
- Entrenamiento de cada clasificador y predicción
- Aplicamos algunas métricas como accuracy, precision, recall y F1-score, macro-F1, weighted-F1 para obtenerlas hemos usado los siguientes comandos en nuestro script:
  - `accuracy_score`
  - `classification_report()`
- Los resultados han sido los siguientes:

```
Accuracy GaussianNB:    0.8575
Accuracy QDA (MVN):     0.90875
Accuracy GMM:           0.9025

=== Report GNB ===
              precision    recall  f1-score   support

        Good       0.95      0.96      0.95       320
   Hazardous       0.72      0.68      0.70        80
    Moderate       0.86      0.90      0.88       240
        Poor       0.73      0.69      0.71       160

    accuracy                           0.86       800
   macro avg       0.81      0.80      0.81       800
weighted avg       0.85      0.86      0.86       800

=== Report QDA (MVN) ===
              precision    recall  f1-score   support

        Good       0.99      0.98      0.99       320
   Hazardous       0.82      0.81      0.82        80
    Moderate       0.89      0.93      0.91       240
        Poor       0.82      0.78      0.79       160

    accuracy                           0.91       800
   macro avg       0.88      0.88      0.88       800
```

```
weighted avg       0.91      0.91      0.91       800

=== Report GMM ===
              precision    recall  f1-score   support

        Good       0.99      0.99      0.99       320
   Hazardous       0.79      0.72      0.76        80
    Moderate       0.91      0.92      0.92       240
        Poor       0.77      0.79      0.78       160

    accuracy                           0.90       800
   macro avg       0.87      0.86      0.86       800
weighted avg       0.90      0.90      0.90       800
```

- Podemos ver que el QDA / MVN supera claramente a GNB y ligeramente al GMM en recall y F1-global
- MVN completo obtiene la mejor accuracy (91%) y el mayor F1-macro (0.88), mostrando un buen balance entre clases mayoritarias y minoritarias
- Captura las correlaciones multivariantes que resultan relevantes para diferencias categorías de calidad del aire.
- Aunque el GMM tiene una ligera ventaja en recall de "poor" no compensa su menos rendimiento global ni la complejidad adicional
- GaussianNB, a pesar de ser robusto, queda notablemente por debajo en clases con menos muestras.
- Conclusión:
  El sistema basado en Multivariante Normal completo (implementado con QDA) es el más adecuado: maximiza la precisión global, mejora la detección de clases minoritarias y mantiene una complejidad razonable.
  Por lo tanto este es el modelo que elegimos para la siguiente parte.

2ª Parte - Implantación e inferencia

1. Importamos las librerías necesarias
2. Cargamos los ficheros `.pkl` con los objetos que guardamos en el anterior apartado
3. Leemos el conjunto de test
4. Aplicamos inferencia (nos devuelve un array con códigos)
5. Conversión de códigos a etiquetas
6. Configuramos el formato de salida pedido en la práctica
7. Al ejecutar este script se nos debería de haber guardado un archivo que se llame `practica_Y_test.csv` con nuestras predicciones

3ª Parte - Clustering

a) Cuantos clusters has elegido? ¿La elección está relacionada con el número de clases? ¿Por qué?

He elegido 4 clusters, porque sabemos a priori que hay 4 categorías de calidad del aire (good, moderate, poor, harzanous). Creo que elegir un número de clusters igual al de clases facilita luego la comparación y el mapeo de cluster a clase

b) Comparación de etiqueta y cluster

La matriz de contingencia muestra cómo se distribuyen las muestras de cada clase real entre los clusters. Si cada cluster está dominado por una única clase, entonces va a existir una buena correspondencia entre cluster y etiqueta. Podemos cuantificarlo con la pureza del cluster

```
Contingency Matrix (cluster vs class):
class    Good  Hazardous  Moderate  Poor
cluster
0         168         40      1195   637
1        1432          0         5     0
2           0        281         0   128
3           0         79         0    35
Cluster→Clase mapping:
{0: 'Moderate', 1: 'Good', 2: 'Hazardous', 3: 'Hazardous'}
```

- **Cluster 1**: casi todas sus 1 437 muestras (1 432) son de la clase **Good** → alta pureza.
- **Cluster 0**: mayoritariamente **Moderate** (1 195), pero un volumen significativo de **Poor** (637) e incluso algo de **Good** y **Hazardous**, lo que indica cierto solapamiento entre "Moderate" y "Poor".
- **Cluster 2**: predominio claro de **Hazardous** (281 vs. 128 de Poor) → se asocia a "Hazardous".
- **Cluster 3**: secundaria agrupación de ejemplos de **Hazardous** (79) y algunos **Poor** (35), revelando que dentro de la clase "Hazardous" puede haber subgrupos o patrones distintos.

Concluimos que:

- **Good** y **Hazardous** se separan muy bien en el espacio preprocesado.
- **Moderate** y **Poor** comparten región y, aunque "Moderate" sale como dominante en cluster 0, más de la mitad de los "Poor" quedan ahí también, lo que sugiere que sus distribuciones son contiguas o tienen solapamiento.

c) Asignación en el test

Cada ejemplo de `X_test` se asigna al cluster cuyo centro está más próximo en el espacio preprocesado. Luego, usando el mapping cluster/clase (determinada en el entrenamiento), asignamos a cada ejemplo la clase mayoritaria de su cluster. El resultado se guarda en `practica3_Y_test.csv`.
Al aplicar el mismo K-Means y el mismo pipeline al conjunto de test:

1. Cada ejemplo de `X_test` recibe un número de cluster (0–3).
2. Usamos el mapeo cluster→clase anterior para convertirlo en etiqueta:
   - Si cae en cluster 1 → "Good"
   - Si cae en cluster 0 → "Moderate"
   - Si cae en cluster 2 o 3 → "Hazardous"

d) ¿Se puede usar clustering para clasificación?

Cada ejemplo de `X_test` se asigna al cluster cuyo centro está más próximo en el espacio preprocesado. Luego, usando el mapping cluster→clase (determinada en el entrenamiento), asignamos a cada ejemplo la clase mayoritaria de su cluster. El resultado se guarda en `practica3_Y_test.csv`.

Aunque en este caso el clustering reproduce razonablemente bien dos de las clases ("Good" y "Hazardous"), la **mezcla** de "Moderate" y "Poor" en el mismo cluster muestra sus límites para clasificación. Por ello, clustering puede ayudar a explorar la estructura de los datos, pero no sustituye a un modelo supervisado cuando las clases se solapan.