Entregable I - Multilabel Datasets

Isabella Hernández García,
z22hegai@uco.es
31 de enero de 2023

ÍNDICE

I. INTRODUCCIÓN

El objetivo de la práctica es familiarizarse con el tema de clasificación multietiqueta utilizando la biblioteca *Scikit-multilearn*[4]

En el *Link a Github* se encuentran implementadas las soluciones a cada uno de los ejercicios.

II. RESPUESTA A LOS EJERCICIOS

En esta sección se da respuesta a cada una de las preguntas que se encontraban explícitas en la orden.

II-A. Ejercicio 2

> *Familiarízate con los datasets presentes en Scikit-multilearn y con los que trabajaremos durante las próximas sesiones. ¿Qué información representan?. Concretamente, dispone de los siguientes 17 datasets:*

```
{'scene', 'Corel5k', 'bibtex',
'enron', 'rcv1subset5', 'tmc2007_500',
'rcv1subset3', 'rcv1subset1', 'delicious',
'rcv1subset4', 'genbase', 'birds',
'emotions','rcv1subset2', 'mediamill',
'medical', 'yeast'}
```

II-A1. Información representada en los datasets:

- Scene: Un conjunto de datos de imágenes con 2407 imágenes etiquetadas hasta en 6 clases: playa, puesta de sol, hojas de otoño, campo, montaña y urbana. Cada imagen se describe con 294 características numéricas visuales correspondientes a momentos de color espacial en el espacio LUV.
- Corel5k: Es una prueba popular para métodos de clasificación y anotación de imágenes. Se basa en 5000 imágenes Corel.

II-B. Ejercicio 3

II-B1. Script car.py: TODO

II-B2. Resumen de las medidas de caracterización: TODO

II-C. Ejercicio 4

> *Busca en Internet dos datasets extra. Descárgalos localmente y realiza un script en Python para cargarlos y recalcular todas las medidas del paso anterior sobre los mismos. Debes escoger datasets variados, con diferente número de etiquetas, variables e instancias.*

La orden no tenía preguntas explícitas para este ejercicio pero se pueden observar los resultados en *Github* TODO

II-D. Ejercicio 5

> *Familiarízate con la documentación scikit-multilearn en: http://scikit.ml/ Prueba los métodos ML disponibles pertenecientes a las dos categorías (transformación y adaptación) que hemos visto en teoría.*

La orden no tenía preguntas explícitas para este ejercicio pero se pueden observar los resultados en *Github*

II-E. Ejercicio 6

II-E1. Ejemplo de código y descripción de cada función :

```
from skmultilearn.dataset import load_dataset

X_train, y_train, _, _ =
↪  load_dataset(set_name="scene", variant="un")
X_test, y_test, _, _ =
↪  load_dataset(set_name="scene", variant="test")
scene:train - exists, not redownloading
scene:test - exists, not redownloading
from skmultilearn.problem_transform import
↪  LabelPowerset
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import hamming_loss
from sklearn.metrics import make_scorer
from sklearn.model_selection import cross_validate,
↪  KFold, cross_val_score
import pandas as pd
from sklearn.metrics import classification_report
classifier = LabelPowerset(
    classifier =
    ↪  RandomForestClassifier(n_estimators=100),
    require_dense = [False, True]
)

# Es una función que crea un objeto que Recibe una
↪  métrica y evalúa el clasificador con ella. Puede
↪  ser utilizado en la evaluación de un modelo con
↪  cross_validate() o cross_val_score() usando el
↪  parámetro scoring.
hamming_scorer = make_scorer(hamming_loss)

# Es una clase que divide un conjunto de datos en
↪  partes para la validación cruzada.
kFold = KFold(n_splits=5)

# Kfold puede ser utilizado para obtener la división
↪  del dataset de la forma kFold.split(X_train) o
↪  puede ser utilizado por cross_validate o por
↪  cross_val_score usando el parámetro cv.

# cross_validate combina la división de datos y la
↪  evaluación del modelo en una sola llamada.
↪  Devuelve tanto las puntuaciones de rendimiento
↪  como los tiempos de entrenamiento y prueba
hamming_scores_with_times =
↪  cross_validate(classifier, X_train, y_train,
↪  scoring=hamming_scorer, cv=kFold)

display(pd
↪  .DataFrame(hamming_scores_with_times).transpose())

# Funciona igual a cross_validate, pero devuelve
↪  solamente la puntuación del modelo en cada fold.
↪  No devuelve los tiempos
hamming_scores = cross_val_score(classifier,
↪  X_train, y_train, scoring=hamming_scorer,
↪  cv=kFold)

display(pd.DataFrame(hamming_scores))
```

	0	1	2	3	4
fit_time	0.902332	0.945678	0.926189	0.879335	0.851875
score_time	0.016609	0.016332	0.016637	0.016828	0.017307
test_score	0.314815	0.157713	0.130165	0.280303	0.307851

Figura 1. Salida de la función cross validate.

	0
0	0.310700
1	0.161846
2	0.127410
3	0.278926
4	0.309229

Figura 2. Salida de la función cross val scores. Solo contiene los resultados de la métrica en cada fold.

II-E2. *Validación cruzada en el contexto de clasificación multi-etiqueta:* TODO

II-F. Ejercicio 7

- $Accuracy$: Es el número de predicciones correctas dividido por el número total de predicciones. En clasificación multietiqueta accuracy se calcula para el subconjunto, y calcula la fracción de veces que todas las etiquetas predichas para una muestra coinciden exactamente con las etiquetas reales. [3]
- $HammingLoss$: Es una métrica de evaluación basada en instancias que evalúa cuántas veces un par instancia-etiqueta se clasifica mal. Hamming Loss penaliza las etiquetas individuales que fueron clasificadas incorrectamente. No tiene en cuenta la relación entre etiquetas correctas e incorrectas. [3]
- $Precision$: Se refiere a la fracción de resultados verdaderos positivos de todos los resultados positivos. Mide la estabilidad de la medida frente a las repeticiones. [3] De todas las predicciones positivas, ¿cuántas son realmente positivas?
- $Recall$: Calcula la fracción de resultados verdaderos positivos que fueron correctamente identificados. [3] De todos los casos positivos reales, ¿cuántos son positivos predichos?
- $F1Score$: $F1Score$ es la media armónica entre la precisión y el recall. Su objetivo sería maximizar tanto la precisión como la recuperación. [3]

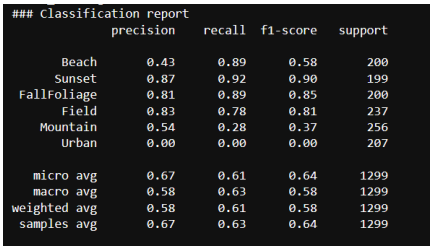
En una tarea de clasificación multiclase y multietiqueta, se pueden aplicar las métricas de precisión, recall y F-measures a cada etiqueta de forma independiente. Hay varias maneras de combinar los resultados a través de las etiquetas, especificadas por el argumento de promedio en las funciones $average\_precision\_score$ (sólo multietiqueta), $f1\_score$, $fbeta\_score$, $precision\_recall\_fscore\_support$, $precision\_score$ y $recall\_score$.

II-G. Ejercicio 8

Escribe un script en Python (cl-cv.py) seleccionando al menos 3 métodos (que NO pertenezcan todos a la misma categoría) para evaluarlos con 5 de los datasets que recopilaste anteriormente y calcula las métricas resultantes mediante validación cruzada. El script mostrará el resultado de las métricas anteriores

II-G1. Script:

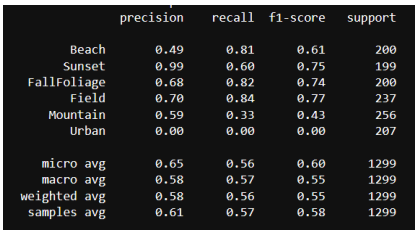
```
from sklearn.metrics import classification_report,
↪  hamming_loss, make_scorer
from sklearn.model_selection import KFold,
↪  cross_val_predict, cross_validate
from skmultilearn.dataset import load_dataset
from utils import get_description
N_SPLITS = 3


def eval_report(classifierFactory, dataset_name):
    X_train, y_train, feature_names, label_names =
    ↪  load_dataset(
        set_name=dataset_name, variant="train")
    X_test, y_test, feature_names, label_names =
    ↪  load_dataset(
        set_name=dataset_name, variant="test")
    target_names = [label_name[0] for label_name in
    ↪  label_names]

    classifier = classifierFactory()

    kFold = KFold(n_splits=N_SPLITS)

    scoring = {**{item: item for item in
    ['accuracy',
     'precision_macro',  'precision_micro',
     'recall_macro',
     'recall_micro',
     'f1_macro',
     'f1_micro']}, **{'hamming':
     ↪  make_scorer(hamming_loss)}}

    scores = cross_validate(classifier, X_train,
    ↪  y_train, scoring=scoring,
                            cv=kFold,
                            ↪  return_train_score=True,
                            ↪  return_estimator=True)

    # Making the classification_report using the
    ↪  test set
    # And returning both cross validation scores and
    ↪  report
    # for comparison
    estimator = scores['estimator'][N_SPLITS - 1]
    y_pred = estimator.predict(X_test)

    report = classification_report(y_test, y_pred,
    ↪  target_names=target_names)

    return scores, report
```



Classification report

	precision	recall	f1-score	support
Beach	0.43	0.89	0.58	200
Sunset	0.87	0.92	0.90	199
FallFoliage	0.81	0.89	0.85	200
Field	0.83	0.78	0.81	237
Mountain	0.54	0.28	0.37	256
Urban	0.00	0.00	0.00	207
micro avg	0.67	0.61	0.64	1299
macro avg	0.58	0.63	0.58	1299
weighted avg	0.58	0.61	0.58	1299
samples avg	0.67	0.63	0.64	1299

Figura 3. Métricas para el dataset Scene con el método Label Powerset with Random Forest



	precision	recall	f1-score	support
Beach	0.49	0.81	0.61	200
Sunset	0.99	0.60	0.75	199
FallFoliage	0.68	0.82	0.74	200
Field	0.70	0.84	0.77	237
Mountain	0.59	0.33	0.43	256
Urban	0.00	0.00	0.00	207
micro avg	0.65	0.56	0.60	1299
macro avg	0.58	0.57	0.55	1299
weighted avg	0.58	0.56	0.55	1299
samples avg	0.61	0.57	0.58	1299

Figura 4. Métricas para el dataset Scene con el método BRkNNa

II-G2. Resultados: En esta sección se describen los resultados obtenidos al aplicar cada uno de los clasificadores a cada dataset

Datasets seleccionados

- scene
- enron
- medical
- genbase
- emotions

Clasificadores seleccionados

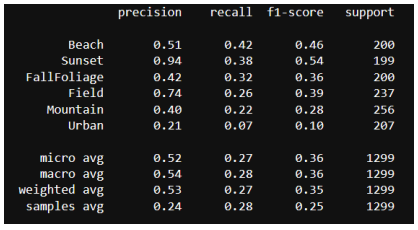
- (Transformation) Label Powerset with Random Forest
- (Adaptation) Binary Relevance kNN classifier (BRkNNa) [2]
- (Adaptation) Multilabel Twin Support Vector Machine (MLTSVM) [1]

III. Conclusiones

Reportar en tercera persona las diferentes conclusiones producto de la práctica de laboratorio desarrollada.

Referencias

[1] Wei-Jie Chen, Yuan-Hai Shao, Chun-Na Li, and Nai-Yang Deng. Mltsvm: a novel twin support vector machine to multi-label learning. *Pattern Recognition*, 52:61–74, 2016.



	precision	recall	f1-score	support
Beach	0.51	0.42	0.46	200
Sunset	0.94	0.38	0.54	199
FallFoliage	0.42	0.32	0.36	200
Field	0.74	0.26	0.39	237
Mountain	0.40	0.22	0.28	256
Urban	0.21	0.07	0.10	207
micro avg	0.52	0.27	0.36	1299
macro avg	0.54	0.28	0.36	1299
weighted avg	0.53	0.27	0.35	1299
samples avg	0.24	0.28	0.25	1299

Figura 5. Métricas para el dataset Scene con el método MLTSVM

[2] Ioannis Vlahavas Eleftherios Spyromitros, Grigorios Tsoumakas. An empirical study of lazy multilabel classification algorithms. In *Proc. 5th Hellenic Conference on Artificial Intelligence (SETN 2008)*, 2008.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] P.ński@ and T. Kajdanowicz. A scikit-based Python environment for performing multi-label classification. *ArXiv e-prints*, February 2017.