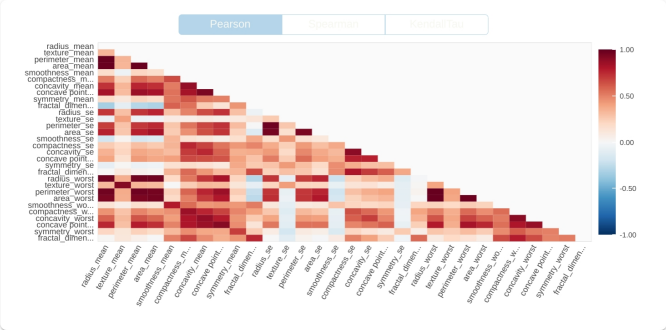
Báo Cáo Đồ Án Cuối Kỳ CS116

Huỳnh Hoàng Vũ - 20520864
CS116.O11.KHTN
Đại học Công nghệ Thông tin



Hình 1. Thông tin độ tương quan giữa các cột sử dụng thư viện dataprep.

I. Giới thiệu bài toán

Bộ dữ liệu Breast Cancer Wisconsin (Diagnostic) [1] là một bộ dữ liệu phổ biến trong lĩnh vực học máy và phân loại. Bài toán liên quan đến việc dự đoán liệu một khối u vú là lành tính (benign) hay ác tính (malignant) dựa trên các đặc trưng y học của khối u.

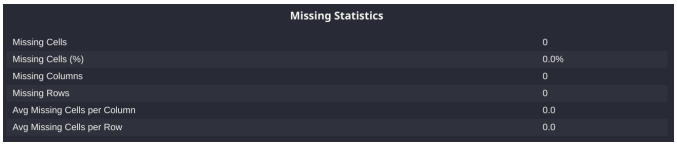
Dữ liệu này chứa thông tin từ các phép đo của những tế bào trong các khối u vú và được sử dụng để xây dựng các mô hình học máy có khả năng dự đoán xác suất của việc u lành tính hoặc ác tính. Mỗi mẫu trong bộ dữ liệu bao gồm các đặc trưng hình ảnh như bán kính, diện tích, độ đo đồng nhất, độ đo độ lệch và nhiều thông số khác.

Đây là một bài toán phân loại hai lớp (binary classification) phổ biến trong lĩnh vực y học và có thể áp dụng nhiều phương pháp học máy như Support Vector Machine (SVM), Decision Tree, Random Forest, Deep Neural Network và các mô hình phân loại khác.

Mục tiêu của bài toán này là xây dựng một mô hình học máy có khả năng dự đoán xác suất hoặc lớp (benign/malignant) của khối u vú dựa trên các đặc trưng hóa học của nó. Điều này có thể giúp các chuyên gia y tế đưa ra quyết định về liệu pháp điều trị và quản lý bệnh nhân.

II. Phân tích dữ liệu khám phá (EDA)

dataprep.eda [2] là một công cụ hữu ích giúp tự động phân tích dữ liệu khám phá (Automated EDA). Với hàm create_report giúp ta có một cái nhìn tổng quan về dữ liệu. Hình 1 và Hình 2 lần lượt thể hiện thông tin độ tương quan giữa các cột và thông tin dữ liệu bị thiếu sử dụng thư viện dataprep. Ta có thể thấy dữ liệu khá đầy đủ khi không có giá trị thiếu nào. Trong khi đó, nhiều có sự tương quan mạnh với nhau.



Hình 2. Thông tin dữ liệu bị thiếu sử dụng thư viện dataprep.

III. Chuẩn bị dữ liệu

Trước hết, cột target "diagnosis" là nhãn cần được phân loại cần được mã hóa từ dạng chuỗi sang dạng số bằng class LabelEncoder của thư viện scikit-learn [3].

```
df[target_col].unique()
```

shape: (2,)

diagnosis
str
"B"
"M"

```
df = df.with_columns(**{
    target_col: LabelEncoder().fit_transform(df[target_col])
})
df[target_col].unique()
```

shape: (2,)

diagnosis
i64
0
1

Sau khi mã hóa, về cơ bản, dữ liệu đã đúng định dạng để đưa vào các mô hình máy học. Em đặt tên bộ dữ liệu sau khi được mã hóa là "orginal". Bên cạnh đó, em còn thực hiện ba phép biến đổi khác từ bộ dữ liệu orginal, được đặt tên lần lượt là "random_corr_remove", "ordered_corr_remove", "pca". Cả random_corr_remove và ordered_corr_remove đều là phép loại bỏ các đặc trưng có độ tương quan cao. Điểm khác nhau duy nhất là random_corr_remove xem xét loại bỏ các đặc trưng theo thứ tự ngẫu nhiên, còn ordered_corr_remove thì ưu tiên loại bỏ các đặc trưng tương quan yếu với target (Hình 3). Với bộ dữ liệu pca, em dùng class PCA của scikit-learn để giảm số đặc trưng của dữ liệu.

IV. Thiết kế thực nghiệm

Để có một kết quả ít bị thiên lệch, em sử dụng 5-fold cross-validation. Thang đo được dùng đến là accuracy.

```diff
- def random_corr_remove(df):
+ def ordered_corr_remove(df):
      global target_col

      correlations = df.corr(method="pearson").abs()
      target_idx = list(df.columns).index(target_col)
-     traversal_order = np.arange(len(df.columns))
-     np.random.seed(random_seed)
-     np.random.shuffle(traversal_order)
+     traversal_order = correlations.loc[target_col].to_numpy().
+     argsort()

      to_remove = []
      for ii, i in enumerate(traversal_order):
          if i == target_idx:
              continue
-         for j in traversal_order[ii + 1 :]:
+         for j in traversal_order[-1:ii:-1]:
              if j == target_idx:
                  continue
              if correlations.iloc[i, j] > threshold:
                  to_remove.append(df.columns[i])
                  break

      return df.drop(columns=to_remove)
```

Hình 3. Sự khác nhau giữa random_corr_remove và ordered_corr_remove trong cài đặt.

```
kf = KFold(n_splits=k, shuffle=True, random_state=random_seed)
```
Python

```
def evaluate(df, model, verbose=0):
    global kf, target_col, tmp_folder, random_seed

    X = df.drop(columns=[target_col]).to_numpy()
    y = df[target_col].to_numpy()

    scores = cross_val_score(deepcopy(model), X, y, cv=deepcopy(kf),
    scoring="accuracy")

    clear_output()
    if verbose > 0:
        print(f"Sores: {scores.round(2)}")
        print(f"Mean accuracy: {np.mean(scores):.4f} +/- {np.std(scores):.4f}")
    return scores.mean()
```
Python

Thực nghiệm tiến hành so sánh thuật toán TabPFN [4] và XGBoost [5] với phiên bản có Cleanlab [6] bọc bên ngoài hoặc không. Đồng thời, so sánh tính hiệu quả của các cách tiền xử lý dữ liệu.

```
eval_models = {
    "tabpfn": TabPFNClassifier(seed=random_seed),
    "xgboost": XGBClassifier(random_state=random_seed),
    "cleanlab + tabpfn": CleanLearning(TabPFNClassifier(seed=random_seed),
    seed=random_seed),
    "cleanlab + xgboost": CleanLearning(XGBClassifier
    (random_state=random_seed), seed=random_seed),
}
```

V. Kết quả thực nghiệm

Dựa vào Bảng I, trên dataframe gốc, ta có thể thấy sử dụng thuật toán XGBoost mang lại độ chính xác cao nhất 97%, kế đến là TabPFN 96%. Đáng ngạc nhiên là ở các dataframe khác TabPFN lại cho độ chính xác cao nhất. Cleanlab không thực sự giúp ích trong thực nghiệm này, khi mà nó làm giảm độ chính xác của thuật toán gốc ở phần lớn các trường hợp (trừ Cleanlab+TabPFN với PCA).

Quan sát các cách tiền xử lý, nhìn chung, PCA mang lại kết quả tệ hơn 2 phép biến đổi còn lại. random_corr_remove hơn điểm dataframe gốc 1 lần, PCA hơn 1 lần, orderd_corr_remove hơn 2 lần cho thấy các cách tiền xử lý trên không thực sự hiệu quả, đặc biệt là với XGBoost.

Khi so sánh 2 kiểu lượt bỏ các đặc trưng tương quan mạnh, 1 là duyệt theo thứ tự ngẫu nhiên random_corr_remove, 2 là duyệt theo thứ tự xác định orderd_corr_remove, orderd_corr_remove mang lại kết quả tốt hơn hẳn. Ta có thể kết luận, việc ưu tiên loại bỏ đặc trưng tương quan yếu với target, giữ lại đặc trưng tương quan mạnh với target là tốt hơn so với việc không quan tâm yếu tố này.

Tài liệu

[1] M. O. S. N. Wolberg, William and W. Street, "Breast Cancer Wisconsin (Diagnostic)," UCI Machine Learning Repository, 1995, DOI: https://doi.org/10.24432/C5DW2B.

[2] J. Peng, W. Wu, B. Lockhart, S. Bian, J. N. Yan, L. Xu, Z. Chi, J. M. Rzeszotarski, and J. Wang, "Dataprep.eda: Task-centric exploratory data analysis for statistical modeling in python," in *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 20–25, 2021, Virtual Event, China*, 2021.

[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*, vol. 12, pp. 2825–2830, 2011.

[4] N. Hollmann, S. Müller, K. Eggensperger, and F. Hutter, "TabPFN: A transformer that solves small tabular classification problems in a second," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=cp5PvcI6w8_

[5] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. ACM, Aug. 2016. [Online]. Available: http://dx.doi.org/10.1145/2939672.2939785

[6] C. G. Northcutt, L. Jiang, and I. L. Chuang, "Confident learning: Estimating uncertainty in dataset labels," *Journal of Artificial Intelligence Research (JAIR)*, vol. 70, pp. 1373–1411, 2021.

	tabpfn	xgboost	cleanlab + tabpfn	cleanlab + xgboost
original	0.962857	**0.974286**	0.948571	0.954286
random_corr_remove	0.96	0.957143	0.954286	0.925714
ordered_corr_remove	0.965714	0.965714	0.957143	0.954286
pca	0.945714	0.937143	0.954286	0.931429

Bảng I
HIỆU SUẤT CỦA CÁC CÁCH TIỀN XỬ LÝ ỨNG VỚI CÁC THUẬT TOÁN.