HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP 1

THỰC TẬP CƠ SỞ

Giáo viên hướng dẫn : Trần Đình Quế

Họ và tên : Đỗ Tuấn Anh

MSV: B22DCCN014

Lớp: D22CQCN02-B

1.Kĩ thuật cleaning data

***Khái niệm*:** Làm sạch dữ liệu hay Data Cleaning là quá trình lấy dữ liệu như bạn hiện có và dọn dẹp nó bằng cách sửa lỗi, xác định các điểm dữ liệu không chính xác, các mục nhập trùng lặp,...

Data Cleaning được xem là một công việc quan trọng không thể thiếu. Quá trình này giúp dữ liệu được chuẩn hóa, tăng tính chính xác, mang đến những câu trả lời đáng tin cậy và giúp các công cụ phân tích dữ liệu truy cập và làm việc thuận lợi.

- Xử lí dữ liệu bị thiếu(Missing Value)
  1. Dữ liệu bị thiếu là một vấn đề phổ biến trong mọi tập dữ liệu. Nếu không xử lý đúng cách, nó có thể ảnh hưởng đến độ chính xác của mô hình.
  2. Loại bỏ các hàng hoặc cột chứa giá trị bị thiếu

3. Phù hợp khi dữ liệu bị thiếu là ngẫu nhiên và chiếm tỷ lệ nhỏ.

- Xử lí dữ liệu bị trùng lặp(Removing Duplicates)
    1. Tìm và loại bỏ các bản ghi trùng lặp trong tập dữ liệu
    2. Có thể dựa trên toàn bộ bản ghi hoặc một số cột cụ thể.
- Dữ liệu không hợp lệ hoặc ngoại lệ(Correcting Inaccurate Data)
    1. Kiểm tra tính hợp lệ của dữ liệu dựa trên các quy tắc và ràng buộc: Chuyển đổi kiểu dữ liệu (string → numeric, date formatting
    2. Xử lý các ký tự đặc biệt không mong muốn
    3. Biểu thức chính quy (regular expressions), so sánh dữ liệu với các nguồn tham khảo.
- Chuẩn hoá dữ liệu(Standardizing Data)
    1. Đảm bảo rằng dữ liệu được định dạng và đo lường một cách nhất quán
    2. Các kỹ thuật chuẩn hóa phổ biến:
       Chuẩn hóa Min-Max: Đưa dữ liệu về dải [0,1], Chuẩn hóa Z-score: Chuyển về phân phối chuẩn ($\mu=0$, $\sigma=1$), Log transformation cho dữ liệu có phân phối lệch
- Xử lý dữ liệu ngoại lai (Handling Outliers)
    1. Xác định và xử lý các giá trị ngoại lai, là những giá trị khác biệt đáng kể so với phần còn lại của dữ liệu.
    2. Phát hiện outliers bằng biểu đồ Box Plot, Z-score, IQR
- Công Cụ Hỗ Trợ
    1. Các thư viện Python: Pandas, NumPy.
    2. Các công cụ phần mềm: OpenRefine, Trifacta Wrangler
    3. Các ngôn ngữ truy vấn cơ sở dữ liệu (SQL).

2.Các kĩ thuật Machine Learning

- ***Thuật toán hồi quy(Regession)***

Thuật toán Hồi quy (Regression) phân loại thuộc nhóm thuật toán Học có giám sát (Dự đoán – Supervised Learning Algorithms). Hồi quy giống như việc ghép một đường thẳng qua các điểm phân tán để đưa ra dự đoán.

Cách thức hoạt động: Nó mô hình hóa mối quan hệ giữa một biến phụ thuộc và một hoặc nhiều biến độc lập bằng cách ghép một đường thẳng hoặc đường cong vào dữ liệu. Mục tiêu là dự đoán kết quả liên tục dựa trên các tính năng đầu vào.

Các loại:

- Hồi quy tuyến tính: Giả định mối quan hệ tuyến tính giữa các biến.
- Hồi quy đa thức: Ghép một đường cong vào dữ liệu.
- Hồi quy logistic: Dự đoán xác suất cho các tác vụ phân loại.

Ứng dụng: Dự đoán giá nhà dựa trên các tính năng, ước tính doanh số dựa trên chi tiêu quảng cáo, dự báo giá cổ phiếu.

- ***Thuật toán K-Nearest Neighbor(KNN)***

  - Thuật toán K-nearest Neighbor (KNN) thuộc phân loại Học có giám sát (Supervised Learning Algorithms) theo phương pháp Phân loại và Hồi quy. KNN giống như việc hỏi những hàng xóm gần gũi và thân cận nhất của bạn lời khuyên. Nó phân loại hoặc dự đoán dựa trên phiếu bầu đa số (để phân loại) hoặc trung bình (để hồi quy) của K điểm dữ liệu gần nhất.

  -Hãy tưởng tượng bạn mới chuyển đến một khu phố và muốn xác định xem bạn có thích sống ở đó hay không dựa trên sở thích của những người hàng xóm.

  Ví dụ 1: Chọn nhà hàng:

  Bạn muốn thử một nhà hàng mới. Bạn hỏi ba người hàng xóm thân thiết nhất của mình để xin lời khuyên. Kết quả là nếu hai người hàng xóm gợi ý đồ ăn Ý và một người gợi ý đồ ăn Trung Quốc, bạn có thể quyết định thử nhà hàng Ý.

  Ví dụ 2: Khuyến nghị phim:

  Bạn không chắc nên xem phim nào. Bạn hỏi năm người bạn thân nhất của mình về những bộ phim yêu thích của họ. Dựa trên lựa chọn của họ, bạn quyết định xem bộ phim mà hầu hết bạn bè của bạn giới thiệu.

- ***Support Vector Machines (SVM)***

  SVM là một thuật toán học máy có giám sát dùng để phân loại và hồi quy, nhưng phổ biến nhất trong bài toán phân loại nhị phân. SVM hoạt động bằng cách tìm một siêu phẳng tối ưu để phân tách các nhóm dữ liệu tốt nhất. Khi dữ liệu không thể phân tách tuyến tính, SVM sử dụng Kernel Trick để ánh xạ dữ liệu sang không gian có nhiều chiều, giúp phân tách dễ hơn. a) SVM với Linear Kernel Linear Kernel là kernel đơn giản nhất trong SVM, giả định rằng dữ liệu có thể được phân tách tuyến tính bằng một siêu phẳng trong không gian đặc trưng ban đầu mà không cần ánh xạ lên không gian chiều cao hơn. Công thức: $K(x, y) = x \cdot y$ Trong đó: + x,y là hai vector đặc trưng. + $x \cdot y$ là tích vô hướng của chúng. Cơ chế hoạt động: Tìm 1 siêu phẳng $w \cdot x + b = 0$ sao cho: + w là vector pháp tuyến của siêu phẳng + b là hằng số bias Khi đã có w và b bằng cách giải bài toán toán tối ưu hóa, SVM dùng siêu phẳng để:

  + Phân loại dữ liệu mới (x trong X_test với X_test là một ma trận, trong đó mỗi hàng là một điểm dữ liệu, và mỗi cột là một đặc trưng): • Tính $w \cdot x + b$. • Nếu > 0: Lớp 1. • Nếu < 0: Lớp -1. + Trong scikit-learn, điều này được thực hiện bằng predict()

b) SVM với Radial Basis Function (RBF) Kernel RBF Kernel (hay Gaussian Kernel) là một kernel phi tuyến, cho phép SVM phân tách dữ liệu phức tạp bằng cách ánh xạ dữ liệu lên không gian chiều cao hơn (thường là vô hạn chiều) thông qua hàm Gaussian. Công thức: $K(x, y) = \exp(-\gamma\|x - y\|2$) Trong đó: + $\|x - y\|2$: Bình phương khoảng cách Euclidean giữa x và y.

+ γ: Tham số điều chỉnh độ rộng của hàm Gaussian ($\gamma = 1\ 2\sigma 2$, với σ là độ lệch chuẩn). Giá trị K(x,y) là giá trị tương đồng giữa x và y nằm trong khoảng [0,1] : + Gần 1 nếu x và y rất gần nhau. + Gần 0 nếu x và y xa nhau. Giá trị K(x,y) thay thế tích vô hướng x·y (như Linear Kernel) trong quá trình tối ưu hóa. Trong dự đoán: $f(x) = \sum wiK(x, xi) + b$ dựa trên K(x, xi) để phân loại mà không cần biết tọa độ trong không gian ánh xạ. Trong đó x là một điểm dữ liệu (từ X_test) và xi là một điểm dữ liệu trong tập huấn luyện (X_train). Nếu $f(x) > 0$ thì dự đoán lớp 1. Nếu $f(x) < 0$ thì dự đoán lớp -1

4. Chạy Case Study, demo, giải thích

- ***Code thuật toán***

[19]:
```
from sklearn import linear_model
from sklearn.model_selection import cross_val_score
df = pd.read_csv('archive\\diabetes.csv')
#---features--
X = df[['Glucose','BMI','Age']]
#---label--
y = df.iloc[:,8]
log_regress = linear_model.LogisticRegression()
log_regress_score = cross_val_score(log_regress, X, y, cv=10,
scoring='accuracy').mean()
print(log_regress_score)
```

```
0.765704032809296
```

[22]:
```
result = []
result.append(log_regress_score)
```

[23]:
```
from sklearn.neighbors import KNeighborsClassifier
#---empty list that will hold cv (cross-validates) scores--
cv_scores = []
#---number of folds--
folds = 10
#---creating odd list of K for KNN--
ks = list(range(1,int(len(X) * ((folds - 1)/folds)), 2))
#---perform k-fold cross validation--
for k in ks:
    knn = KNeighborsClassifier(n_neighbors=k)
    score = cross_val_score(knn, X, y, cv=folds, scoring='accuracy').mean()
    cv_scores.append(score)
 #---get the maximum score--
knn_score = max(cv_scores)
#---find the optimal k that gives the highest score--
optimal_k = ks[cv_scores.index(knn_score)]
print(f"The optimal number of neighbors is {optimal_k}")
print(knn_score)
result.append(knn_score)
```

```
The optimal number of neighbors is 19
0.7669514695830485
```

[24]:
```
from sklearn import svm
linear_svm = svm.SVC(kernel='linear')
linear_svm_score = cross_val_score(linear_svm, X, y,
                                   cv=10, scoring='accuracy').mean()
print(linear_svm_score)
result.append(linear_svm_score)
```

```
0.7630724538619276
```

[26]:
```
rbf = svm.SVC(kernel='rbf')
rbf_score = cross_val_score(rbf, X, y, cv=10, scoring='accuracy').mean()
print(rbf_score)
result.append(rbf_score)
```

```
0.7617908407382091
```

Giải thích:

1. Đọc dữ liệu
   - Chương trình đọc dữ liệu từ tệp CSV (diabetes.csv) và chọn các đặc trưng (Glucose, BMI, Age) làm biến đầu vào (X).
   - Nhãn đầu ra (y) được lấy từ cột thứ 8 của DataFrame.
2. Logistic Regression
   - Áp dụng Logistic Regression và đánh giá mô hình bằng cross-validation với cv=10.
   - In ra độ chính xác trung bình (log_regress_score).
3. K-Nearest Neighbors (KNN)
   - Duyệt qua các giá trị k (số hàng xóm) lẻ.
   - Với mỗi k, thực hiện cross-validation để tính độ chính xác.
   - Lưu lại giá trị k tối ưu cho mô hình KNN (optimal_k).
   - Lưu điểm số cao nhất (knn_score) vào danh sách result.
4. Support Vector Machine (SVM)
   - Chạy SVM với kernel tuyến tính (linear), đánh giá bằng cross-validation và lưu kết quả vào result.
   - Chạy SVM với kernel RBF (rbf), đánh giá bằng cross-validation và lưu kết quả vào result.
5. Tổng hợp kết quả
   - Lưu độ chính xác của từng thuật toán vào một DataFrame.
   - Sắp xếp theo độ chính xác giảm dần để xem mô hình nào hoạt động tốt nhất.

Kết Luận

- Cleaning data quan trọng để đảm bảo dữ liệu đồng nhất và chính xác.
- Random Forest cho hiệu suất tốt nhất trong case study.
- Có thể cải thiện mô hình bằng tuning tham số (GridSearchCV).