عنوان پروژه:

پیش‌بینی بیماری قلبی با استفاده از الگوریتم Logistic Regression و مقایسه روش‌هایEncoding

مخزن پروژه:

https://github.com/Nargesazizollahi/heart-disease-logistic-regression?utm_source=chatgpt.com

چکیده

در این پروژه، یک مدل یادگیری ماشین برای تشخیص بیماری قلبی با استفاده از الگوریتم Logistic Regression پیاده‌سازی شده است. هدف اصلی، بررسی تأثیر دو روش مختلف تبدیل ویژگی categorical شامل One-Hot Encoding و Label Encoding بر عملکرد مدل می‌باشد. متغیر هدف دیتاست که در ابتدا دارای پنج سطح شدت بیماری بود، به یک مسئله طبقه‌بندی دودویی تبدیل شد (وجود یا عدم وجود بیماری). برای ارزیابی عملکرد مدل از روش 10-Fold Cross Validation استفاده شد و معیارهایAccuracy ، Precision، Recallو F1-score گزارش گردیدند. نتایج نشان داد که روش One-Hot Encoding به دلیل جلوگیری از القای ترتیب مصنوعی بین دسته‌ها، عملکرد بهتری به‌ویژه در معیار Recall و F1-score ارائه می‌دهد. با توجه به اهمیت کاهش خطای منفی کاذب در کاربردهای پزشکی، One-Hot Encodingگزینه مناسب‌تری برای این مسئله تشخیص داده شد.

۱.مقدمه

هدف این پروژه، پیاده‌سازی یک مدل یادگیری ماشین برای تشخیص بیماری قلبی با استفاده از الگوریتم Logistic Regression است. علاوه بر آموزش مدل، تأثیر دو روش مختلف تبدیل متغیر categorical شامل One-Hot Encoding و Label Encoding بر عملکرد مدل مورد بررسی قرار می‌گیرد.

مسئله به صورت طبقه‌بندی دودویی تعریف شده است (وجود یا عدم وجود بیماری). برای ارزیابی عملکرد مدل از روش ۱۰-Fold Cross Validation و معیارهایAccuracy ، Precision، Recallو F1-score استفاده شده است.

۲.آماده سازی داده ها

۲-۱ معرفی دیتاست

در این پروژه از دیتاست Heart Disease استفاده شده است که شامل ویژگی‌های بالینی بیماران بوده و متغیر هدف آن با نام num مشخص می‌شود.

در دیتاست اصلی، متغیر numدارای پنج مقدار مختلف است:

- 0 : عدم وجود بیماری
- 1تا ۴ : وجود بیماری با شدت‌های مختلف

از آنجا که هدف پروژه طبقه‌بندی دودویی است، این متغیر به دو کلاس تبدیل شد که در بخش پیاده‌سازی کد توضیح داده می‌شود.

$$num = 0 \rightarrow -1$$
$$num = 1,2,3,4 \rightarrow +1$$

۲-۲ بررسی توزیع کلاس‌ها

پس از تبدیل متغیر هدف به حالت دودویی، توزیع کلاس‌ها بررسی شد.

```
print("Class distribution after binarization:")
print(df['num'].value_counts())
```

```
Class distribution after binarization:
num
 1    509
-1    411
```

شکل ۱ – توزیع کلاس‌ها پس از دودویی‌سازی متغیر هدف

همان‌طور که در شکل ۱ مشاهده می‌شود، تعداد نمونه‌های کلاس مثبت ۵۰۹ و کلاس منفی ۴۱۱ است. اگرچه اختلاف جزئی وجود دارد، اما داده‌ها نسبتاً متوازن هستند. با این حال در تحلیل عملکرد مدل توجه ویژه‌ای به معیار Recall شده است.

۲-۳ تبدیل متغیر هدف به طبقه‌بندی دودویی

در دیتاست اصلی، متغیر هدف numدارای پنج سطح مختلف (۰ تا ۴) است. از آنجا که هدف پروژه طراحی یک مدل طبقه‌بندی دودویی است، این متغیر به دو کلاس تبدیل شد:

- مقدار ۰ : کلاس -1 عدم وجود بیماری
- مقادیر ۱ تا ۴ : کلاس +1 وجود بیماری

```
df['num'] = df['num'].apply(lambda x: -1 if x == 0 else 1)
```

df['num'] ستون هدف را انتخاب می‌کند.

تابع apply() برای اعمال یک تابع روی تمام مقادیر ستون استفاده می‌شود.

عبارت lambda x: -1 if x == 0 else 1 یک تابع ناشناس است که:

- اگر مقدار برابر ۰ باشد، مقدار -۱ برمی‌گرداند.
- در غیر این صورت مقدار +۱ اختصاص می‌دهد.

این مرحله باعث تبدیل مسئله از حالت چندکلاسه به طبقه‌بندی دودویی شد.

۳. پیاده‌سازی مدل

۳-۱ وارد کردن کتابخانه

```
import pandas as pd
import numpy as np

from pathlib import Path
from sklearn.impute import SimpleImputer
from sklearn.model_selection import KFold, cross_validate
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
```

- pandas : برای مدیریت داده‌ها.
- numpy : برای محاسبات عددی مثل میانگین و انحراف معیار.
- KFold : برای اعتبارسنجی متقاطع ۱۰ بخشی.
- cross_validate : برای محاسبه همزمان چند معیار رویK-Fold .
- Pipeline : برای اتصال مراحل پیش‌پردازش و مدل‌سازی.
- ColumnTransformer : برای اعمال پیش‌پردازش متفاوت روی ستون‌های عددی وcategorical .
- OneHotEncoderو LabelEncoder : برای دو روش encoding ستون cp .
- StandardScaler : برای استانداردسازی ویژگی‌های عددی.
- SimpleImputer :برای جایگزینی مقادیر گمشده داخل Pipeline .
- LogisticRegression :مدل اصلی طبقه بندی.

۳-۲ بارگذاری داده

در این مرحله فایل دیتاست از پوشه /dataخوانده می‌شود:

```
df = pd.read_csv("data/heart_disease_uci.csv")
```

این دستور فایل CSV را خوانده و آن را در قالب یک DataFrame در متغیر dfذخیره می‌کند. دیتافریم ساختاری جدولی دارد که امکان انجام پردازش‌های بعدی روی داده را فراهم می‌کند.

۳-۳ حذف برخی ویژگی ها

در این مرحله برخی از ویژگی‌های غیرپیوسته یا کم‌اهمیت حذف شدند تا تمرکز پروژه بر مقایسه روش‌های Encoding برای ویژگی cp باشد.

```
drop_cols = ['sex', 'dataset', 'fbs', 'restecg', 'exang', 'slope', 'thal']
df = df.drop(columns=drop_cols)
```

در خط اول، یک لیست به نام drop_cols تعریف شده که شامل نام ستون‌هایی است که باید حذف شوند.
در خط دوم، با استفاده از تابع ()drop این ستون‌ها از دیتافریم حذف می‌شوند.
پارامتر =columns مشخص می‌کند که حذف بر اساس نام ستون انجام شود.
نتیجه دوباره در df ذخیره شده است.
هدف از این کار ساده‌سازی مدل و باقی ماندن تنها یک ویژگی categorical (cp) برای مقایسه دو روش Encoding بوده است.

۳-۴ جداسازی ویژگی ها و متغیر هدف

```
X = df.drop(columns=['num'])
y = df['num']
```

در خط اول، ستون هدف (num) حذف شده و باقی ستون‌ها به عنوان ویژگی‌ها در متغیر Xذخیره می‌شوند.
در خط دوم، ستون num به عنوان متغیر هدف در yذخیره می‌شود.
این جداسازی در مسائل یادگیری نظارتی ضروری است.

۳-۵ تعیین ویژگی categorical و عددی

```
categorical_feature = ['cp']
numerical_features = [col for col in X.columns if col not in categorical_feature]
```

در خط اول مشخص شده که تنها ویژگی categorical برابر با cp است.
در خط دوم با استفاده از List Comprehension تمامی ستون‌هایی که در لیست categorical نیستند، به عنوان ویژگی عددی در نظر گرفته می‌شوند.
این تقسیم‌بندی برای اعمال پیش‌پردازش متفاوت ضروری است.

۴. طراحی Pipeline برای One-Hot Encoding

۴-۱ پیش‌پردازش ویژگی‌های عددی

```
num_pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])
```

Pipeline برای اتصال چند مرحله پردازش به صورت زنجیره‌ای استفاده می‌شود.
مرحله اول:

- SimpleImputer(strategy='median')
- مقادیر گمشده عددی را با میانه جایگزین می‌کند.
- استفاده از median به دلیل مقاومت در برابر داده‌های پرت است.

مرحله دوم:

- StandardScaler()
- داده‌ها را استاندارد می‌کند (میانگین صفر و واریانس یک).
- این کار برای مدل‌های خطی مانند Logistic Regression اهمیت دارد.

۴-۲ ترکیب پیش‌پردازش عددی وcategorical

```
preprocess_onehot = ColumnTransformer([
    ('num', num_pipeline, numerical_features),
    ('cat', OneHotEncoder(drop='first'), categorical_feature)
])
```

ColumnTransformer اجازه می‌دهد روی ستون‌های مختلف پردازش متفاوت اعمال شود.

بخش اول:

- 'num'نام دلخواه این مرحله است.
- num_pipelineروی ویژگی‌های عددی اعمال می‌شود.

بخش دوم:

- 'cat'نام دلخواه مرحله encoding است.
- OneHotEncoder(drop='first') روی ستون cp اعمال می‌شود.
- گزینه drop='first'برای جلوگیری از Dummy Variable Trap استفاده شده است (جلوگیری از هم‌خطی کامل بین ویژگی‌ها).

۴-۳ ساخت مدل نهاییOne-Hot

```
model_onehot = Pipeline([
    ('preprocess', preprocess_onehot),
    ('classifier', LogisticRegression(max_iter=1000))
])
```

در این Pipeline دو مرحله وجود دارد:

۱. preprocess شامل تمام مراحل آماده‌سازی داده

۲. classifier مدلLogistic Regression

پارامتر max_iter=1000 برای اطمینان از همگرایی کامل الگوریتم تنظیم شده است.

استفاده از Pipeline تضمین می‌کند که در هر Fold از Cross Validation:

- مراحلImputation ، Scalingو Encoding فقط روی داده‌های آموزش fit (Train) شوند.
- سپس همان تبدیل‌ها روی داده‌های آزمون (Test) اعمال شوند.

این موضوع از بروز Data Leakageجلوگیری می‌کند و ارزیابی علمی و صحیحی فراهم می‌سازد.

۵.طراحی Pipeline برایLabel Encoding

در این بخش، ستون categorical یعنی cpبه جای One-Hot به روش Label Encoding به عدد تبدیل می‌شود تا اثر نوع Encodingبر عملکرد مدل مقایسه گردد.

۵-۱ اعمال Label Encoding روی ویژگیcp

```
X_label = X.copy()
le = LabelEncoder()
X_label['cp'] = le.fit_transform(X_label['cp'])
```

X.copy() یک کپی مستقل از داده‌ها می‌سازد تا تغییرات روی X_label باعث تغییر در Xاصلی نشود.
LabelEncoder() یک شیء برای تبدیل مقادیر categorical به عدد ایجاد می‌کند.
fit_transform() دو کار انجام می‌دهد:

۱. fitمقادیر منحصربه‌فرد cpرا یاد می‌گیرد.
۲. transformهر مقدار را به یک عدد صحیح نگاشت می‌کند.

در نهایت، ستون cpدر X_labelعددی می‌شود.

در Label Encoding ممکن است مدل به اشتباه ترتیب عددی بین دسته‌ها را واقعی فرض کند) مثلاً cp=3 بزرگ‌تر از(cp=1 ، در حالی که cpیک ویژگی Nominalاست و ترتیب واقعی ندارد. به همین دلیل ممکن است عملکرد کاهش یابد.

۵-۲ پیش‌پردازش داده‌ها در حالتLabel Encoding

```
preprocess_label = ColumnTransformer([
    ('num', num_pipeline, X_label.columns)
])
```

از آنجا که cpدر این حالت به عدد تبدیل شده، دیگر categorical محسوب نمی‌شود . بنابراین تمام ستون‌ها (شامل cp و سایر ویژگی‌ها) وارد num_pipelineمی‌شوند.
num_pipelineشامل:

- جایگزینی مقادیر گمشده با median
- استانداردسازی با StandardScaler

این کار باعث می‌شود تنها تفاوت دو مدل در نوع Encoding باشد و بقیه مراحل کاملاً یکسان باقی بمانند.

۵-۳ ساخت مدل نهاییLabel Encoding

```
model_label = Pipeline([
    ('preprocess', preprocess_label),
    ('classifier', LogisticRegression(max_iter=1000))
])
```

ساختار Pipeline مشابه مدل One-Hot است تا مقایسه منصفانه انجام شود.
تفاوت اصلی فقط در نحوه نمایش ستون cp است.
max_iter=1000 برای جلوگیری از عدم همگرایی الگوریتم تنظیم شده است.

۶. تعریف معیارهای ارزیابی وCross Validation

۶-۱ تعریف معیارها(Scoring)

```
scoring = {
    'accuracy': 'accuracy',
    'precision': 'precision',
    'recall': 'recall',
    'f1': 'f1'
}
```

یک دیکشنری به نام scoringتعریف شده است تا چند معیار به صورت همزمان محاسبه شوند.
معیارها:

- Accuracy:درصد پیش‌بینی درست کلی
- Precision:از بین پیش‌بینی‌های مثبت، چند درصد واقعاً مثبت بوده‌اند
- Recall:از بین مثبت‌های واقعی، چند درصد درست شناسایی شده‌اند (مهم‌ترین معیار در پزشکی)
- F1-score:میانگین هماهنگ Precision وRecall

در مسائل پزشکی Recall بسیار مهم است، چون خطای False Negative(بیمار واقعی که سالم تشخیص داده شود) خطرناک‌تر است.

۶-۲ تعریف10-Fold Cross Validation

```
kf = KFold(n_splits=10, shuffle=True, random_state=42)
```

n_splits=10 یعنی داده به ۱۰ بخش تقسیم می‌شود.
shuffle=True باعث می‌شود قبل از تقسیم‌بندی، داده‌ها تصادفی مخلوط شوند.
random_state=42 باعث می‌شود نتایج قابل تکرار (Reproducible) باشند.
استفاده از تنظیمات ثابت برای هر دو مدل باعث می‌شود مقایسه دو روش Encoding کاملاً منصفانه باشد.

۷.اجرای ارزیابی مدل ها

۷-۱ اجرای Cross Validation برای مدلOne-Hot

```
scores_onehot = cross_validate(model_onehot, X, y, cv=kf, scoring=scoring)
print("One-Hot Encoding Results:", scores_onehot)
```

cross_validate() مدل را با استفاده از 10 -Fold Cross Validation ارزیابی می‌کند.

پارامترها:

- model_onehot مدل شامل پیش‌پردازش+Logistic Regression
- X ویژگی‌ها
- y متغیر هدف
- cv=kf شیء تعریف‌شدهKFold
- scoring=scoring معیارهای ارزیابی

خروجی این تابع یک دیکشنری شامل آرایه‌ای از نتایج هر معیار برای هر Fold است.

دستور printبرای مشاهده جزئیات کامل نتایج هر Fold استفاده شده است.

۷-۲ اجرای Cross Validation برای مدلLabel Encoding

```
scores_label = cross_validate(model_label, X_label, y, cv=kf, scoring=scoring)
print("Label Encoding Results:", scores_label)
```

۸. خلاصه‌سازی نتایج(Mean ± Std)

۸-۱ تعریف تابع خلاصه سازی

```
def print_summary(scores, title):
    print(f"\n{title} - Mean ± Std")
    for metric in ['test_accuracy', 'test_precision', 'test_recall', 'test_f1']:
        mean = np.mean(scores[metric])
        std = np.std(scores[metric])
        name = metric.replace('test_', '').capitalize()
        print(f"{name:<10}: {mean:.3f} ± {std:.3f}")
```

یک تابع به نام print_summaryتعریف شده که:

- ورودی اول :scores (خروجیcross_validate)
- ورودی دوم: title (عنوان مدل)

در حلقه for، هر معیار ارزیابی بررسی می‌شود.

()np.mean میانگین عملکرد در ۱۰ Fold را محاسبه می‌کند.

()np.std میزان نوسان عملکرد (پایداری مدل) را محاسبه می‌کند.

('test_', '')replace برای خواناتر شدن نام معیار استفاده شده است.

خروجی به صورت Mean ± Stdچاپ می‌شود.

گزارش Mean ± Std نشان می‌دهد مدل نه‌تنها چه عملکردی دارد، بلکه چقدر پایدار است و به یک تقسیم‌بندی خاص وابسته نیست.

۸-۲ چاپ نهایی

```
print("\n=============== SUMMARY ===============")
print_summary(scores_onehot, "One-Hot Encoding")
print_summary(scores_label, "Label Encoding")
print("=======================================\n")
```

ابتدا یک خط جداکننده چاپ می‌شود. سپس نتایج مدل One-Hot خلاصه می‌شود. بعد نتایج مدل Label Encoding چاپ می‌شود. در نهایت خروجی مرتب و قابل ارائه تولید می‌شود.

۹.تحلیل نتایج

نتایج به‌دست‌آمده نشان داد که روش One-Hot Encoding در اکثر معیارها به‌ویژه Recall و F1-score عملکرد بهتری نسبت به Label Encodingدارد.

این موضوع از نظر تئوری نیز قابل توجیه است؛ زیرا:

- ویژگی cpماهیت Nominalدارد و ترتیب واقعی بین دسته‌ها وجود ندارد.
- Label Encodingممکن است به مدل القا کند که بین مقادیر رابطه ترتیبی وجود دارد.
- One-Hot Encodingاین مشکل را برطرف می‌کند و هر دسته را به صورت مستقل نمایش می‌دهد.

از منظر کاربرد پزشکی، معیار Recall اهمیت بیشتری دارد، زیرا کاهش خطای منفی کاذب (بیمار واقعی که سالم تشخیص داده شود) حیاتی است. بنابراین با توجه به Recall بالاتر، روش One-Hot Encoding انتخاب مناسب‌تری برای این مسئله محسوب می‌شود.

۹-۱ نتایج عددی

در این بخش، عملکرد دو روش Encoding با استفاده از Fold-10 Cross Validation-مقایسه شده است.

جدول ۱- مقایسه عملکرد دو روش کدگذاری(Mean ± Std)

📈 **Results (10-Fold Cross Validation)**

The model was evaluated using 10-fold cross validation. Missing values were handled using `SimpleImputer` inside the pipeline to avoid data leakage.

Encoding Method	Accuracy (Mean ± Std)	Precision (Mean ± Std)	Recall (Mean ± Std)	F1-score (Mean ± Std)
One-Hot Encoding	**0.809 ± 0.048**	0.822 ± 0.053	**0.833 ± 0.081**	**0.825 ± 0.055**
Label Encoding	0.803 ± 0.047	**0.822 ± 0.048**	0.818 ± 0.090	0.818 ± 0.059

✅ **Conclusion:** One-Hot Encoding achieved slightly better overall performance, especially in Recall and F1-score.

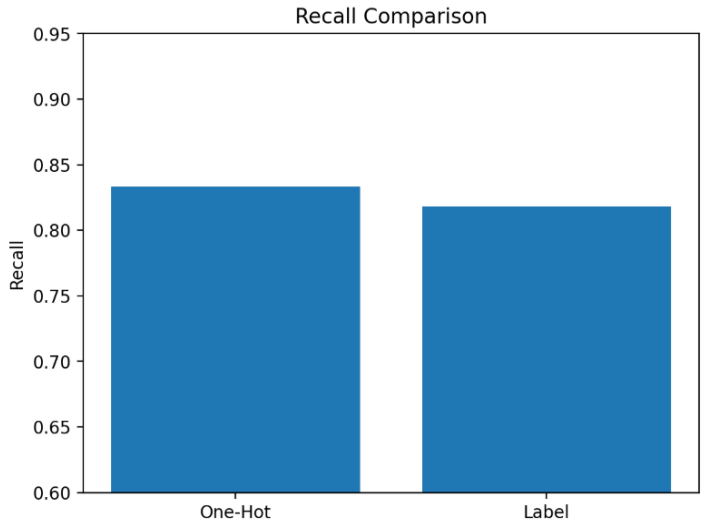
همان‌طور که در جدول ۱ مشاهده می‌شود، روش One-Hot Encoding در معیارهای Recall و F1-score عملکرد بهتری ارائه داده است.

۹-۲ نمایش گرافیکی نتایج

```
15
16  ================ SUMMARY ================
17
18  One-Hot Encoding - Mean ± Std
19  Accuracy  : 0.809 ± 0.048
20  Precision : 0.822 ± 0.053
21  Recall    : 0.833 ± 0.081
22  F1        : 0.825 ± 0.055
23
24  Label Encoding - Mean ± Std
25  Accuracy  : 0.803 ± 0.047
26  Precision : 0.822 ± 0.048
27  Recall    : 0.818 ± 0.090
28  F1        : 0.818 ± 0.059
29  =========================================
30
```

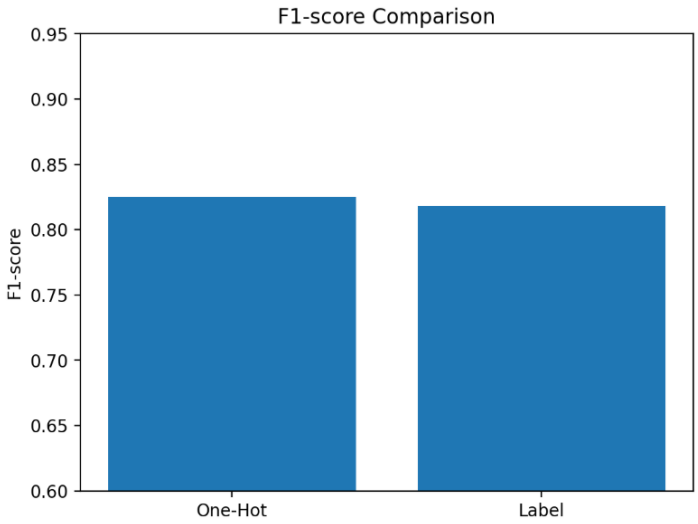
شکل۲- نتایج اعتبارسنجی متقاطع ۱۰ بخشی برای دو روشEncoding

به‌منظور درک بهتر تفاوت عملکرد دو روشEncoding ، مقایسه گرافیکی معیارهای Recall و F1-score ارائه شده است.



شکل ۳ – مقایسه Recall برای دو روشEncoding

همان‌طور که در شکل ۳ مشاهده می‌شود، مقدار Recall در روش One-Hot Encoding بالاتر از Label Encoding است. این موضوع نشان می‌دهد که مدل در حالت One-Hot توانایی بیشتری در شناسایی بیماران واقعی (کلاس مثبت +۱) دارد و احتمال خطای منفی کاذب کاهش یافته است. با توجه به اهمیت تشخیص صحیح بیماران در کاربردهای پزشکی، این اختلاف هرچند اندک، از نظر عملی حائز اهمیت است.



شکل ۴ – مقایسه F1-score برای دو روشEncoding

مطابق شکل ۴، مقدار F1-score در روش One-Hot Encoding نیز اندکی بالاتر از Label Encoding است. این نتیجه نشان می‌دهد که علاوه بر افزایشRecall ، تعادل کلی میان Precision و Recall نیز در روش One-Hot بهتر حفظ شده است. بنابراین، از منظر عملکرد کلی مدل، One-Hot Encodingانتخاب مناسب‌تری برای این مسئله محسوب می‌شود.

۱۰. نتیجه گیری

در این پروژه، الگوریتم Logistic Regression برای تشخیص بیماری قلبی پیاده‌سازی شد و تأثیر دو روش Encoding مختلف مورد مقایسه قرار گرفت. ارزیابی با استفاده از 10-Fold Cross Validationانجام شد و معیارهایAccuracy ، Precision، Recallو F1-scoreگزارش گردید.

نتایج نشان داد که One-Hot Encoding عملکرد بهتری ارائه می‌دهد، به‌ویژه در معیار Recall که در کاربردهای پزشکی اهمیت بالایی دارد. بنابراین برای ویژگی‌های categorical از نوعNominal ، استفاده از One-Hot Encoding توصیه می‌شود.

۱۱. ساختار پروژه و سازمان‌دهی فایل‌ها

در این پروژه، ساختار پوشه‌ها و فایل‌ها مطابق نسخه موجود در مخزن GitHub به‌صورت ماژولار طراحی شده است تا جداسازی مناسبی بین بخش‌های مختلف پیاده‌سازی ایجاد شود.

ساختار پوشه‌ها به شرح زیر است:

- **data/** : شامل فایل دیتاست اصلی(heart_disease_uci.csv)
- **figures/** : شامل نمودارهای تولیدشده برای مقایسه عملکرد مدل‌ها
- **report/** : شامل فایل گزارش نهایی پروژه
- **src/** : شامل فایل‌های اصلی پیاده‌سازی مدل

در پوشه srcدو فایل اصلی وجود دارد:

۱. heart_disease_logistic_regression.py

این فایل شامل مراحل اصلی پروژه است:

- بارگذاری داده‌ها
- پیش‌پردازش (حذف ویژگی‌ها، مدیریت مقادیر گمشده، نرمال‌سازی)
- پیاده‌سازی دو روش One-Hot Encoding وLabel Encoding
- ساختPipeline
- اجرای ۱۰-Fold Cross Validation
- محاسبه و چاپ معیارهای ارزیابی

۲. plot_results.py

این فایل مسئول رسم نمودارهای مقایسه‌ای بین دو روش Encoding بوده و نتایج مربوط به معیارهایAccuracy ، Precision، Recallو F1-score را به‌صورت گرافیکی نمایش می‌دهد.

این طراحی ماژولار باعث جداسازی منطق آموزش مدل از بخش مصورسازی شده و خوانایی، نگهداری و توسعه کد را بهبود می‌بخشد.