سوال ۱: پیش‌پردازش داده و طبقه‌بندی باSVM

هدف تمرین

در این تمرین شما با یک دیتاست پزشکی کار خواهید کرد که هدف آن پیش‌بینی ابتلا به دیابت در زنان قبیله پیما است. این تمرین سه مهارت کلیدی را پوشش می‌دهد:

۱. **مدیریت داده‌های گمشده** (Missing Value Imputation)
۲. **مهندسی ویژگی** (Feature Engineering)
۳. **طبقه‌بندی با SVM** (Support Vector Machine)

معرفی دیتاست:

دیتاست در لینک زیر قرار گرفته است:

https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

دیتاست شامل ۷۶۸ رکورد و ۹ ستون است. ستون‌ها عبارتند از:

Pregnancies: تعداد دفعات بارداری (عدد صحیح)

Glucose: غلظت گلوکز خون به mg/dL

BloodPressure: فشار خون دیاستولیک به mm Hg

SkinThickness: ضخامت چربی زیر پوست بازو به mm

Insulin: میزان انسولین سرم خون به mu U/ml

BMI: شاخص توده بدنی به kg/m^2

DiabetesPedigreeFunction: سابقه خانوادگی دیابت (عدد اعشاری)

Age: سن به سال

Outcome: نتیجه تشخیص که ۰ یعنی سالم و ۱ یعنی دیابتی

نکته مهم:در این دیتاست، **مقادیر صفر** در برخی ستون‌ها به معنای **داده گمشده** هستند! از نظر پزشکی غیرممکن است که این مقادیر صفر باشند:

- Glucose = 0 ❌ (غیرممکن)
- BloodPressure = 0 ❌ (غیرممکن)
- SkinThickness = 0 ❌ (غیرممکن)
- Insulin = 0 ❌ (غیرممکن)
- BMI = 0 ❌ (غیرممکن)

بخش اول Missing Value Imputation :

قسمت الف: بارگذاری دیتاست

دیتاست را با استفاده از pandas بارگذاری کنید. از کتابخانه‌های زیر استفاده خواهید کرد:

- pandas برای کار با داده
- numpy برای محاسبات عددی
- matplotlib و seaborn برای رسم نمودار

در این قسمت باید انجام دهید:

اول: پنج سطر اول دیتاست را نمایش دهید

دوم: اطلاعات کلی شامل shape و dtypes را نمایش دهید

سوم: آمار توصیفی با دستور describe را نمایش دهید

قسمت ب: شناسایی Missing Values پنهان

مقادیر صفر در پنج ستون زیر در واقع داده‌های گمشده هستند:

- Glucose
- BloodPressure
- SkinThickness
- Insulin
- BMI

در این قسمت باید کد بنویسید که:

اول: تعداد مقادیر صفر در هر ستون را محاسبه کند

دوم: درصد Missing را برای هر ستون محاسبه کند

قسمت دو: تبدیل صفرها به NaN

قبل از شروع Imputation، ابتدا باید صفرها را به NaN تبدیل کنید. برای این کار از دستور replace استفاده کنید و مقدار صفر را با np.nan جایگزین کنید.

پس از تبدیل، با استفاده از دستور isnull تایید کنید که تعداد NaN در هر ستون چقدر است.

قسمت سه: استراتژی‌های Imputation

شما باید سه استراتژی مختلف را پیاده‌سازی و مقایسه کنید.

استراتژی A: میانگین ساده

هدف: جایگزین کردن مقادیر گمشده با میانگین کل

گام‌های اجرا:

۱. یک کپی از DataFrame اصلی بسازید و نامش را df_strategy_A بگذارید

۲. از SimpleImputer با 'strategy='mean استفاده کنید

. پنج ستون دارای Missing را Transform کنید

۴. بررسی کنید که دیگر NaN نداریم

سوال: میانگین ستون Glucose قبل و بعد از Imputation را محاسبه کنید. چرا تغییر کرد؟

استراتژی B: میانگین گروهی

هدف: جایگزین کردن Missing با میانگین گروه (بر اساس Outcome)

منطق: میانگین Glucose در افراد سالم با دیابتی‌ها فرق دارد!

گام‌های اجرا:

۱. یک کپی به نام df_strategy_B بسازید

۲. برای هر ستون دارای Missing:

- میانگین افراد سالم (Outcome=0) را حساب کنید
- میانگین افراد دیابتی (Outcome=1) را حساب کنید
- Missing های افراد سالم را با میانگین سالم‌ها پر کنید
- Missing های افراد دیابتی را با میانگین دیابتی‌ها پر کنید

۳.این کار را برای چهار ستون دیگر هم تکرار کنید

سوال.میانگین Glucose را برای هر دو گروه چاپ کنید. تفاوت چقدر است؟

استراتژی C: KNN Imputation

هدف: استفاده از ۵ همسایه نزدیک برای پیش‌بینی مقدار گمشده

گام‌های اجرا:

۱. یک کپی به نام df_strategy_C بسازید

۲. از KNNImputer با n_neighbors=5 استفاده کنید.

۳. پنج ستون دارای Missing را Transform کنید

۴. بررسی کنید که دیگر NaN نداریم

سوال: KNN چطور کار می‌کند؟ چرا ممکن است بهتر از میانگین ساده باشد؟

بخش دوم: Feature Engineering

در این بخش، شما باید **۵ ویژگی جدید** بسازید که بر اساس دانش پزشکی، به مدل کمک می‌کنند تا بهتر تشخیص دهد فرد دیابت دارد یا خیر.

نکته مهم: از این به بعد، فقط روی **یکی از سه استراتژی Imputation** کار کنید که در بخش اول انتخاب کرده‌اید (مثلاً df_strategy_B). در انتها می‌توانید ببینید کدام استراتژی بهترین نتیجه را داده است.

ویژگی ۱: دسته‌بندی سطح قند خون

هدف: تبدیل ستون Glucose به دسته‌های پزشکی معنادار.

استاندارد پزشکی:

- کمتر از ۱۰۰: طبیعی (Normal)
- ۱۰۰ تا ۱۲۵: پیش‌دیابت (Prediabetes)
- بیشتر از ۱۲۵: دیابتی (Diabetic)

سوال تحلیلی: یک نمودار ستونی رسم کنید که نشان دهد در هر دسته از Glucose_Category، چند درصد از افراد مبتلا به دیابت هستند. آیا الگوی مشخصی مشاهده می‌شود؟

ویژگی ۲: دسته‌بندی شاخص توده بدنی

هدف: تبدیل ستون BMI به دسته‌های استاندارد وزنی.

استاندارد WHO (سازمان جهانی بهداشت):

- کمتر از ۱۸.۵: کم‌وزن (Underweight)
- ۱۸.۵ تا ۲۵: طبیعی (Normal)
- ۲۵ تا ۳۰: اضافه‌وزن (Overweight)
- بیشتر از ۳۰: چاق (Obese)

سوال تحلیلی: کدام دسته از BMI_Category بیشترین نرخ ابتلا به دیابت را دارد؟ این یافته با دانش پزشکی شما مطابقت دارد؟

ویژگی ۳: گروه‌بندی سنی

هدف: تقسیم ستون Age به ۴ گروه سنی.

دسته‌بندی سنی:

- ۲۱ تا ۳۰ سال: جوان (Young)
- ۳۱ تا ۴۵ سال: میانسال (Middle_Aged)
- ۴۶ تا ۶۰ سال: بالای میانسال (Senior)
- بیشتر از ۶۰ سال: سالمند (Elderly)

سوال تحلیلی: آیا با افزایش سن، نرخ ابتلا به دیابت افزایش می‌یابد؟ از داده‌ها برای پاسخ استفاده کنید.

ویژگی ۴: نسبت انسولین به قند خون :

هدف: ساخت یک ویژگی عددی جدید از ترکیب دو ستون موجود.

منطق پزشکی: نسبت انسولین به گلوکز می‌تواند نشانه‌ای از مقاومت انسولینی (Insulin Resistance) باشد که یکی از عوامل کلیدی در دیابت نوع ۲ است.

```
df['Insulin_to_Glucose_Ratio'] = df['Insulin'] / df['Glucose']
```

- همبستگی (correlation) این ویژگی جدید با Outcomeرا محاسبه کنید .
- آیا این ویژگی به نظر می‌رسد برای پیش‌بینی دیابت مفید باشد؟ چرا؟

ویژگی ۵: وضعیت فشار خون

هدف: دسته‌بندی ستون BloodPressure به وضعیت‌های پزشکی.

استاندارد پزشکی (فشار خون دیاستولیک):

- کمتر از ۸۰: پایین (Low)
- ۸۰ تا ۹۰: طبیعی (Normal)
- بیشتر از ۹۰: بالا (High)

بخش سوم: آموزش و ارزیابی SVM

مرحله اول: آماده‌سازی داده‌ها

۱. One-Hot Encoding

تمام ویژگی‌های دسته‌بندی شده (categorical) باید به فرمت عددی تبدیل شوند. از آنجایی که این ویژگی‌ها ماهیت ترتیبی (ordinal) ندارند، از **One-Hot Encoding** استفاده می‌کنیم.

۲. مقیاس‌بندی داده‌ها (Scaling)

نکته بسیار مهم: الگوریتم SVM به مقیاس ویژگی‌ها بسیار حساس است.

توجه :فقط ویژگی‌های عددی پیوسته را استاندارد کنید، **نه ویژگی‌های One-Hot شده** که مقادیر ۰ و ۱ دارند!

۴ .تقسیم داده به Train وTest

سوال مفهومی :چرا از پارامتر stratify=yاستفاده کردیم؟ این کار چه تضمینی برای ما فراهم می‌کند؟

مرحله دوم: آموزش SVM با سه Kernel مختلف

توضیح کلی:

الگوریتم SVM می‌تواند از توابع مختلفی به نام **Kernel** برای نگاشت داده‌ها به فضای با ابعاد بالاتر استفاده کند. ما سه Kernel محبوب را امتحان می‌کنیم تا ببینیم کدام یک برای این مسئله بهتر عمل می‌کند:

۱. **Linear Kernel**: وقتی داده‌ها به صورت خطی قابل جداسازی هستند
۲. **RBF Kernel**: برای الگوهای غیرخطی پیچیده (محبوب‌ترین)
۳. **Polynomial Kernel**: برای الگوهای چندجمله‌ای

Kernel.1 اول: Linear

توضیح:

Kernel خطی فرض می‌کند که یک **خط مستقیم** (یا ابرصفحه‌ای در فضای چندبعدی) می‌تواند دو کلاس (دیابتی و غیردیابتی) را از هم جدا کند. این ساده‌ترین حالت است و زمانی مناسب است که رابطه بین ویژگی‌ها و برچسب خطی باشد.

Kernel.2 دوم RBF - Radial Basis Function :

توضیح:

Kernel RBF می‌تواند الگوهای غیرخطی بسیار پیچیده را یاد بگیرد. این محبوب‌ترین Kernel برای مسائل واقعی است.

Kernel.3 سوم: Polynomial

توضیح:

Kernel چندجمله‌ای می‌تواند روابط چندجمله‌ای بین ویژگی‌ها را مدل کند. ما از degree=3 استفاده می‌کنیم.

مرحله سوم: ارزیابی و مقایسه مدل‌ها

۱. معیارهای ارزیابی

توضیح:

برای ارزیابی مدل‌های طبقه‌بندی، از چهار معیار اصلی استفاده می‌کنیم:

الف) Accuracy (دقت کلی)
ب) Precision (دقت مثبت)

از بین کسانی که مدل گفته دیابتی هستند، چند نفر واقعاً دیابتی بودند

ج) Recall (بازیابی یا حساسیت)

از بین کسانی که واقعاً دیابتی هستند، مدل چند نفرشان را پیدا کرده

د) F1-Score (میانگین هارمونیک)

میانگین متوازنی از Precision و Recall:\

سوال :اگر تشخیص غلط دیابت (False Negative) از نظر پزشکی خطرناک‌تر از تشخیص اشتباه فرد سالم به عنوان دیابتی (False Positive) باشد، کدام معیار Precision یا Recall برای ما مهم‌تر است؟

۲.Confusion Matrix :

- از نتایج Confusion Matrix سه مدل، کدام مدل کمترین False Negative (افراد دیابتی که به اشتباه سالم تشخیص داده شدند) را دارد؟
- این موضوع از نظر کاربرد پزشکی چه اهمیتی دارد؟

سوال تحلیلی :کدام Kernel بهترین عملکرد را داشته است؟ دلیل برتری آن چه می‌تواند باشد؟

تکمیل با استراتژی‌های Imputation

دستورالعمل نهایی:

تمام مراحل بالا را برای **هر سه استراتژی Imputation** (A، B، C) تکرار کنید

سوال تحلیلی نهایی :

- کدام **ترکیب** از (استراتژی Imputation + Kernel) بهترین F1-Score را داده است؟
- چرا این ترکیب عملکرد بهتری داشته؟ (حداقل دو دلیل ذکر کنید)
- آیا یک الگوی مشخص وجود دارد؟ مثلاً آیا یک استراتژی Imputation خاص در تمام Kernelها بهتر عمل کرده است؟