Institute for Advanced Studies
in Basic Sciences
GavaZang, Zanjan, Iran

Machine Learning Regression, Classification, and Clustering Project

---

Prof. Mahmoud Shirazi

Report for Project <3>

Faeze Ahmadi

14044141

November 2025

پروژه ماشین لرنینگ، رگرسیون، کلاس‌بندی و خوشه بندی

پروژه‌ای که برای تمرین سوم انجام شد، در واقع یک تمرین کامل و مرحله به مرحله برای آشنایی عملی با سه مفهوم اصلی در یادگیری ماشین است: رگرسیون، کلاس‌بندی و خوشه‌بندی. هدف اصلی این پروژه این بود که ما بتوانیم با داده‌های واقعی کار کنیم، آن‌ها را آماده‌سازی کنیم، مدل مناسب بسازیم، مدل را آموزش بدهیم، عملکرد آن را ارزیابی کنیم و در نهایت خروجی مدل را به شکل قابل فهم و قابل نمایش ارائه کنیم.

این پروژه تلاش می‌کند تمام مسیر معمولی که یک مسئله یادگیری ماشین طی می‌کند را شبیه‌سازی کند؛ یعنی از مرحله آماده‌سازی داده گرفته تا مرحله تحلیل نتایج و ذخیره‌سازی مدل. به همین دلیل، در سه بخش مختلف طراحی شده که هر کدام یک نوع مسئله متداول در یادگیری ماشین را پوشش می‌دهد. در بخش اول که مربوط به رگرسیون است، هدف پیش‌بینی مقدارهای پیوسته است؛ مثلا تخمین قیمت خانه بر اساس ویژگی‌هایی مثل تراکم جمعیت، سن ساختمان و مقدار اتاق‌ها. این نوع مسائل در دنیای واقعی بسیار رایج هستند و تقریبا در هر صنعتی که بخواهیم مقدار مشخصی را پیش‌بینی کنیم، مثلا قیمت، دما، سود، مصرف انرژی، و ... همین نوع مدل‌ها استفاده می‌شوند.

در بخش دوم وارد کلاس‌بندی یا Classification می‌شویم؛ مسئله‌ای که به جای پیش‌بینی یک عدد، باید بگوییم داده به کدام دسته تعلق دارد. مثال معروف آن تشخیص نوع گل‌های Iris است که به خاطر سادگی و ساختار مناسبش در درس‌ها و پژوهش‌ها زیاد استفاده می‌شود. با این بخش یاد می‌گیریم مدل چطور الگوهای بین ویژگی‌ها و برچسب‌ها را یاد می‌گیرد، چگونه دقت مدل سنجیده می‌شود و چطور نتایج مدل را با استفاده از ابزارهایی مثل ماتریس درهم‌ریختگی تفسیر کنیم.

در بخش سوم به سراغ خوشه‌بندی می‌رویم که یکی از شاخه‌های یادگیری بدون نظارت (Unsupervised Learning) است. در اینجا هیچ برچسبی وجود ندارد و مدل باید خودش از روی ساختار داده حدس بزند که

داده‌ها به چند دسته تقسیم می‌شوند. این روش در بسیاری از کارهای واقعی مانند بخش‌بندی مشتریان، تشخیص گروه‌های رفتاری، یا تحلیل شباهت داده‌ها کاربرد دارد. در این پروژه از الگوریتم KMeans استفاده شده و برای قابل مشاهده کردن نتایج، داده‌ها به کمک روش کاهش بعد PCA به دو بعد تبدیل شده‌اند تا خوشه‌ها روی نمودار نمایش داده شوند.

در نسخه پیشرفته، علاوه بر نسخه ساده هر بخش، سعی شده پروژه به شکل حرفه‌ای‌تر و واقعی‌تر پیاده‌سازی شود؛ یعنی با استفاده از شی‌گرایی (OOP)، ثبت گزارش‌ها (Logging)، مدیریت خطاها (Error Handling)، ذخیره مدل‌ها برای استفاده مجدد و همچنین رسم نمودارهای دقیق‌تر و زیباتر. این باعث شده پروژه از حالت یک تکلیف ساده خارج شود و به شکل یک مینی‌پروژه قابل ارائه و استانداردتر تبدیل شود؛ چیزی که در پروژه‌های صنعتی و پژوهشی هم کاربرد دارد.

به طور خلاصه، این پروژه یک تمرین عملی است تا ما به عنوان دانشجو با سه خانواده اصلی مدل‌های یادگیری ماشین آشنا شویم، تفاوت آن‌ها را درک کنیم، نحوه کار با داده‌ها و ارزیابی مدل‌ها را یاد بگیریم و در نهایت بتوانیم یک پروژه کوچک اما حرفه‌ای را از صفر تا صد پیاده‌سازی کنیم.

Github Link: https://github.com/Faeze-Ahmadi/advanced-programming-assignment-3/tree/main/part1_regression

توضیح کلی پارت 1 – Regression

در بخش اوّل پروژه، هدف ما این است که با استفاده از یادگیری ماشین، «قیمت تقریبی خانه‌ها در ایالت کالیفرنیا» را پیش‌بینی کنیم. این کار یک مثال استاندارد از مسئله‌ی رگرسیون است؛ یعنی خروجی مدل یک عددِ پیوسته است، نه یک برچسب دسته‌بندی. ما عملاً به کامپیوتر یک سری ویژگی عددی درباره‌ی هر ناحیه مسکونی می‌دهیم (مثل میانگین سن ساختمان‌ها، تراکم جمعیت، میانگین اتاق‌ها، درآمد متوسط و ...) و از آن می‌خواهیم حدس بزند که «قیمت متوسط خانه» در آن ناحیه چقدر است.

این قسمت از پروژه دو لایه دارد: یک نسخه‌ی ساده و آموزشی، و یک نسخه‌ی پیشرفته‌تر که به‌صورت شی‌گرا، همراه با ثبت گزارش (logging)، مدیریت خطا، ذخیره‌ی مدل و رسم نمودار حرفه‌ای نوشته شده است. در ادامه توضیح می‌دهم که در این پارت از چه کتابخانه‌هایی استفاده کرده‌ایم، هرکدام چرا لازم بوده‌اند، و بعد قدم‌به‌قدم منطق کدها را تشریح می‌کنم.

کتابخانه‌ها و ابزارهایی که در پارت 1 استفاده می‌کنیم

در نسخه ساده و نسخه پیشرفته تقریباً از یک مجموعه ابزار استفاده می‌شود، با این تفاوت که در نسخه‌ی پیشرفته ابزارهای بیشتری برای حرفه‌ای شدن پروژه اضافه شده است.

هسته‌ی کار با کتابخانه‌ی scikit-learn است. از زیرماژول datasets تابعی به نام fetch_california_housing استفاده می‌کنیم. این تابع یک دیتاست آماده‌ی استاندارد را از روی اینترنت (یا کش محلی) بارگذاری می‌کند که شامل اطلاعات واقعی بازار مسکن کالیفرنیاست. به کمک این تابع دیگر نیازی نیست خودمان فایل CSV را جداگانه دانلود و پردازش کنیم؛ تمام اطلاعات به صورت آماده و ساختارمند به ما داده می‌شود.

برای تقسیم داده‌ها به بخش آموزش و آزمون از train_test_split در ماژول model_selection استفاده می‌کنیم. این تابع ورودی‌ها و خروجی‌ها را طوری تقسیم می‌کند که بخشی برای آموزش مدل استفاده شود و بخشی برای ارزیابی، تا بفهمیم مدل روی داده‌های جدید چقدر خوب عمل می‌کند.

برای خودِ مدل رگرسیون از LinearRegression در ماژول linear_model استفاده شده است. این کلاس همان مدل رگرسیون خطی کلاسیک است که سعی می‌کند یک رابطه‌ی خطی بین ویژگی‌ها و قیمت خانه پیدا کند؛ یعنی ضرایب بهینه‌ای پیدا می‌کند که خط یا صفحه‌ی مناسب‌تری روی داده‌ها قرار بگیرد.

برای رسم نمودارها از matplotlib.pyplot استفاده می‌کنیم. این ماژول به ما اجازه می‌دهد نمودار پراکندگی بین مقادیر واقعی و مقادیر پیش‌بینی‌شده را رسم کنیم تا به صورت بصری ببینیم مدل چقدر خوب کار کرده است.

در نسخه‌ی پیشرفته چند کتابخانه‌ی دیگر هم اضافه شده‌اند. ماژول os برای ساختن پوشه‌ی خروجی و کنار هم نگه داشتن فایل‌ها استفاده می‌شود؛ به کمک آن مطمئن می‌شویم اگر پوشه‌ی پارت ۱ وجود نداشت، قبل از ذخیره تصویر و مدل ساخته شود. ماژول logging برای ثبت پیام‌های اطلاعاتی و خطا به‌صورت استاندارد استفاده شده

است. به‌جای print ساده، با logging مشخص می‌کنیم که هر پیام از چه نوعی است (اطلاع، خطا و ...) و خروجی برنامه خواناتر و حرفه‌ای‌تر می‌شود. کتابخانه‌ی joblib برای ذخیره‌سازی مدل آموزش‌دیده در فایل pkl به کار می‌رود؛ یعنی می‌توانیم مدل را یک‌بار آموزش دهیم و بعداً بدون آموزش مجدد، آن را از روی فایل بخوانیم و استفاده کنیم. ماژول numpy (با نام کوتاه np) هم برای محاسبه‌ی خط برازش روی نقاط و رسم خط رگرسیون روی نمودار استفاده شده است. در نهایت، از NotFittedError در sklearn.exceptions استفاده می‌شود تا اگر کسی قبل از آموزش مدل، از آن بخواهد پیش‌بینی یا ارزیابی انجام دهد، بتوانیم این وضعیت را درست مدیریت کنیم.

Github Link: https://github.com/Faeze-Ahmadi/advanced-programming-assignment-3/blob/main/part1_regression/regression.py

توضیح نسخه ساده پارت 1 – Regression

در نسخه‌ی ساده، روند کار کاملاً خطی است و همه‌چیز پشت سر هم در بدنه‌ی اصلی فایل نوشته شده است. ابتدا دیتاست بارگذاری می‌شود؛ شیء برگردانده‌شده شامل چند بخش است. مهم‌ترین بخش‌ها data (همان ویژگی‌ها) و target (قیمت خانه) هستند. این دو را در متغیرهای X و y قرار می‌دهیم تا از نظر خوانایی کد واضح‌تر شوند؛ X نماینده‌ی ماتریس ویژگی‌هاست و y بردار قیمت‌ها.

در گام بعد از train_test_split استفاده می‌کنیم تا داده‌ها را دو قسمت کنیم: X_train و y_train برای آموزش، X_test و y_test برای آزمون. معمولاً درصد ثابتی (مثلاً ۲۰ درصد) را برای تست کنار می‌گذاریم و یک random_state ثابت انتخاب می‌کنیم تا هر بار اجرای کد، تقسیم‌بندی یکسانی انجام شود و نتایج قابل تکرار باشند.

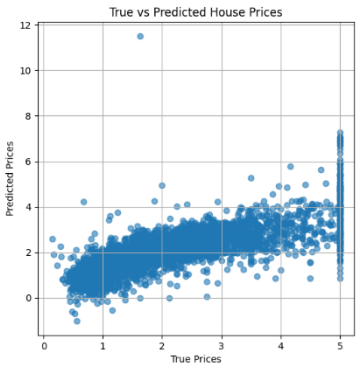
بعد یک شیء از کلاس LinearRegression می‌سازیم و با فراخوانی متد fit روی داده‌های آموزش، مدل را آموزش می‌دهیم. این متد سعی می‌کند ضرایب بهینه‌ی رگرسیون را به دست بیاورد. بعد از این مرحله مدل آماده‌ی پیش‌بینی است. با متد predict روی X_test، مقادیر y_pred به‌عنوان قیمت‌های پیش‌بینی‌شده به دست می‌آیند.

برای ارزیابی مدل از متد score استفاده می‌کنیم که در رگرسیون خطی به صورت پیش‌فرض مقدار ضریب تعیین (R^2) را محاسبه می‌کند. این عدد بین منفی بی‌نهایت و یک است که هرچه به یک نزدیک‌تر باشد، مدل بهتر به داده‌ها می‌خورد. این مقدار روی ترمینال چاپ می‌شود تا به صورت عددی کیفیت مدل را ببینیم.

در نهایت یک نمودار رسم می‌کنیم که روی محور افقی قیمت‌های واقعی (y_test) و روی محور عمودی قیمت‌های پیش‌بینی‌شده (y_pred) قرار دارند. با plt.scatter نقاط را رسم می‌کنیم، عنوان و برچسب محورها را تنظیم می‌کنیم و شبکه‌ی زمینه (grid) را فعال می‌کنیم تا خواناتر شود. سپس نمودار در قالب فایل PNG داخل پوشه‌ی پارت ۱ ذخیره می‌شود و با plt.show() روی صفحه نمایش داده می‌شود.

این نسخه ساده همه‌ی مراحل اصلی یک مسئله‌ی رگرسیون را پوشش می‌دهد: بارگذاری داده، تقسیم، آموزش، ارزیابی و مصورسازی.

Output: True vs Predicted Chart



این نمودار رابطه بین قیمت‌های واقعی و قیمت‌های پیش‌بینی شده توسط مدل رگرسیون را نشان می‌دهد. اگر مدل به خوبی یاد گرفته باشد، نقاط باید نزدیک به یک خط قطری قرار بگیرند. هر چه پراکندگی کمتر باشد، یعنی مدل در پیش‌بینی قیمت خانه‌ها عملکرد بهتری داشته است.

Github Link: https://github.com/Faeze-Ahmadi/advanced-programming-assignment-3/blob/main/part1_regression/regression_advanced.py

توضیح نسخه پیشرفته و شیء گرای پارت 1 – Regression

در نسخه پیشرفته، همین منطق را به شکل یک کلاس به نام HousingRegressor درآورده‌ایم تا کد ساختارمندتر، قابل توسعه‌تر و حرفه‌ای‌تر باشد. به‌جای اینکه همه‌چیز پشت سر هم در سطح بالا نوشته شود، هر مسئولیت به یک متد اختصاص داده شده است.

در سازنده‌ی کلاس، یک نمونه از LinearRegression ساخته می‌شود، مسیر ذخیره‌سازی (مثلاً part1_regression) در یک ویژگی نگه داشته می‌شود و اگر این پوشه وجود نداشته باشد با استفاده از os.makedirs ساخته می‌شود. همچنین با یک پیام logging.info اعلام می‌شود که شیء رگرسور ساخته شده است.

متد load_data مسئول خواندن دیتاست است. درون این متد، fetch_california_housing فراخوانی می‌شود، X و y استخراج می‌شوند و در عین حال نام ویژگی‌ها برای استفاده‌های احتمالی بعدی ذخیره می‌شود. این عملیات داخل یک بلوک try/except قرار دارد تا اگر مشکلی در بارگذاری داده به وجود آمد (مثلاً قطع اینترنت در اولین اجرا)، پیام خطای مناسب ثبت شود و خطا به بالا پاس داده شود.

متد split_data داده‌ها را با استفاده از train_test_split به چهار بخش استاندارد آموزش و آزمون تقسیم می‌کند. باز هم هرگونه خطای احتمالی در بلوک try/except گرفته و با پیام مناسب logging.error گزارش می‌شود.

متد train وظیفه‌ی آموزش مدل را دارد. در این متد متغیرهای X_train و y_train ورودی گرفته می‌شوند و روی self.model.fit اعمال می‌شوند. در صورت موفقیت، یک پیام اطلاعاتی ثبت می‌شود که «مدل با موفقیت آموزش داده شد». اگر هرگونه خطایی در این مرحله رخ دهد (مثلاً ابعاد داده‌ها ناسازگار باشند)، خطا ثبت و دوباره پرتاب می‌شود.

متد evaluate برای محاسبه‌ی همان R^2 است، اما با مدیریت خطا. در این متد، self.model.score روی داده‌های آزمون اجرا می‌شود و عدد به‌دست‌آمده هم در لاگ ثبت می‌شود و هم به عنوان خروجی متد بازگردانده می‌شود. اگر مدل قبل از این مرحله آموزش داده نشده باشد، NotFittedError رخ می‌دهد؛ در این صورت پیام مناسب در لاگ ثبت می‌شود و خطا دوباره پرتاب می‌شود تا توسعه‌دهنده بداند ترتیب اجرای کد را رعایت نکرده است.

متد predict هم مشابه است و روی self.model.predict یک لایه‌ی مدیریت خطا قرار می‌دهد تا مطمئن شویم قبل از هر پیش‌بینی، مدل آموزش دیده است.

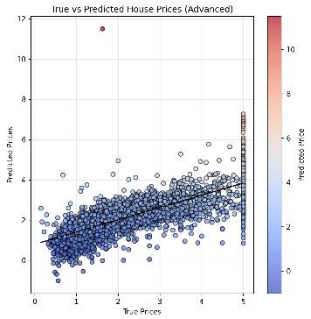
متد save_model مدل آموزش‌دیده را با استفاده از joblib.dump در فایلی با پسوند pkl ذخیره می‌کند. مسیر فایل با os.path.join ساخته می‌شود تا مستقل از سیستم‌عامل درست کار کند. بعد از ذخیره، با یک پیام در لاگ اطلاع داده می‌شود که مدل در چه مسیری ذخیره شده است.

متد plot_results مسئول رسم نمودار پیشرفته‌ی نتیجه است. این متد دو آرایه‌ی y_true و y_pred را می‌گیرد. یک شکل مربع با اندازه‌ی مناسب ساخته می‌شود، نقاط با plt.scatter رسم می‌شوند، رنگ نقاط بر اساس مقدار پیش‌بینی‌شده تنظیم می‌شود (با یک colormap زیبا مثل coolwarm) و برای حاشیه‌ی نقاط از رنگ مشکی استفاده می‌شود تا نقاط مشخص‌تر باشند. سپس plt.colorbar اضافه می‌شود تا مقدار رنگ‌ها معنی داشته باشد و مشخص شود هر رنگ چه بازه‌ای از قیمت را نشان می‌دهد. برچسب محورها و عنوان نمودار تنظیم می‌شوند و شبکه‌ی پس‌زمینه با شفافیت کم فعال می‌شود.

یک گام مهم دیگر، محاسبه‌ی یک خط برازش روی نقاط است. با numpy.polyfit یک خط روی رابطه‌ی بین y_true و y_pred به‌صورت کمترین مربعات برازش می‌دهیم و با numpy.poly1d تابع خط را می‌سازیم. سپس این خط به صورت یک خط چین مشکی روی نمودار رسم می‌شود تا بهتر بفهمیم پیش‌بینی‌ها حول چه روندی توزیع شده‌اند. در پایان، تصویر با وضوح بالا (dpi=300) در مسیر مناسب ذخیره می‌شود و با plt.show نمایش داده می‌شود. پیام نهایی نیز در لاگ ثبت می‌کند که خروجی نمودار در چه مسیری قرار گرفته است.

در انتهای فایل، یک بلوک "__main__" == __name__ if__": قرار دارد که ورودیِ اصلی برنامه است. این بخش کاری شبیه سناریوی زیر انجام می‌دهد: ابتدا یک شیء از HousingRegressor می‌سازد، سپس داده را بارگذاری می‌کند، آن را تقسیم می‌کند، مدل را آموزش می‌دهد، ارزیابی می‌کند، پیش‌بینی می‌گیرد، مدل را ذخیره می‌کند و در آخر نمودار پیشرفته را رسم و ذخیره می‌کند. تمام این مراحل با پیام‌های logging همراهی می‌شوند، بنابراین کاربر یا استاد با نگاه به خروجی ترمینال می‌تواند بفهمد که هر مرحله دقیقاً چه زمانی انجام شده و آیا مشکلی پیش آمده یا نه.

Output: Advanced Regression Chart (OOP Version)



در نسخه پیشرفته، علاوه‌بر نقاط واقعی، یک خط برازش خطی نیز رسم شده است که روند کلی پیش‌بینی‌ها را بهتر نشان می‌دهد. وجود نوار رنگی (Colorbar) کمک می‌کند مقدارها بهتر تفسیر شوند و دیده شود که مدل در چه بازه‌هایی دقیق‌تر است. این نمودار نمای حرفه‌ای‌تری از عملکرد مدل ارائه می‌دهد.

Github Link: http://github.com/Faeze-Ahmadi/advanced-programming-assignment-3/blob/main/part2_logistic_regression/classification.py

توضیح نسخه ساده پارت 2 – Classification با Logistic Regression

در بخش دوم پروژه، هدف این بود که یک مسئله طبقه‌بندی را با استفاده از داده‌های واقعی حل کنیم. برای این کار از دیتاست مشهور Iris استفاده شد؛ دیتاستی که سال‌هاست در حوزه هوش مصنوعی برای آموزش مفاهیم اولیه‌ی classification استفاده می‌شود. ایده‌ی اصلی این بخش این است که مدل یاد بگیرد ویژگی‌های یک گل (مثل طول و عرض گلبرگ و کاسبرگ) را بگیرد و تشخیص بدهد که این گل متعلق به کدام گونه است.

در ابتدای کد، کتابخانه‌هایی را وارد کردیم که هرکدام نقش خاصی در پروژه دارند.

ابتدا از matplotlib.pyplot استفاده می‌کنیم تا در انتهای کار بتوانیم داده‌ها را رسم کنیم. دلیل نیاز به این کتابخانه این است که classification فقط یک عدد دقت نمی‌خواهد؛ باید بتوانیم رفتار مدل را از طریق نمودار ببینیم. رسم نقاط و رنگ‌کردن آن‌ها با برچسب‌های مدل کمک می‌کند بفهمیم آیا مدل داده‌ها را درست جدا کرده یا خیر.

سپس pandas را وارد کردیم. دلیل استفاده از pandas این است که کار با دیتاست‌ها در قالب DataFrame بسیار ساده‌تر و قابل‌فهم‌تر از آرایه‌های خام numpy است. pandas به ما کمک می‌کند ستون‌ها را با اسمشان مدیریت کنیم و داده‌ها را خواناتر ببینیم.

کتابخانه‌ی بعدی، یعنی load_iris از sklearn.datasets، همان ابزاری است که دیتاست Iris را برای ما بارگذاری می‌کند. این دیتاست به‌صورت داخلی در sklearn وجود دارد و نیازی به دانلود فایل جداگانه نیست. این ابزار، داده‌ها را همراه با اسامی ویژگی‌ها برمی‌گرداند و باعث می‌شود بتوانیم نسبت به ستون‌ها آگاهی داشته باشیم.

بعد از بارگذاری داده، آن را داخل یک DataFrame ریختیم. ستون‌های ویژگی (مثل طول کاسبرگ) داخل DataFrame قرار گرفتند و یک ستون جدید به نام target اضافه شد که مقدار آن همان کلاس گل‌هاست (0، 1 یا 2). این کار باعث می‌شود تفکیک ویژگی‌ها از برچسب‌ها کاملاً روشن باشد.

سپس داده را به دو بخش train و test تقسیم کردیم. این مرحله بسیار مهم است، چون اگر تمام داده‌ها را برای آموزش مدل استفاده کنیم، نمی‌توانیم بفهمیم مدل روی داده‌های جدید چقدر خوب عمل می‌کند. با استفاده از train_test_split این جداسازی به‌صورت تصادفی اما کنترل‌شده انجام می‌شود.

گزینه‌ی stratify=y هم به این دلیل اضافه شده تا نسبت کلاس‌ها در مجموعه‌ی آموزش و تست برابر بماند؛ چون اگر مثلاً یک کلاس کمتر در تست قرار بگیرد، مدل نمی‌تواند ارزیابی درستی انجام دهد.

بعد از آماده‌سازی داده‌ها، مدل Logistic Regression ساخته و آموزش داده شد. Logistic Regression یکی از ساده‌ترین و درعین‌حال مهم‌ترین مدل‌های classification است. این مدل سعی می‌کند مرزبندی بین کلاس‌ها را به صورت یک رابطه ریاضی پیدا کند و مکان نقاط را در فضای ویژگی‌ها تحلیل کند.

گزینه‌ی max_iter=300 هم به این دلیل گذاشته شده که مدل در بعضی مواقع نیاز به تکرارهای بیشتر برای همگرایی دارد. پس از آموزش مدل، داده‌های test را به مدل دادیم و خروجی پیش‌بینی‌شده را دریافت کردیم. سپس مقدار accuracy محاسبه شد که نشان می‌دهد مدل روی داده‌های جدید با چه دقتی عملکرد دارد. در این پروژه accuracy معمولاً حدود 94 درصد تا 97 درصد است که مقدار بسیار خوبی محسوب می‌شود.

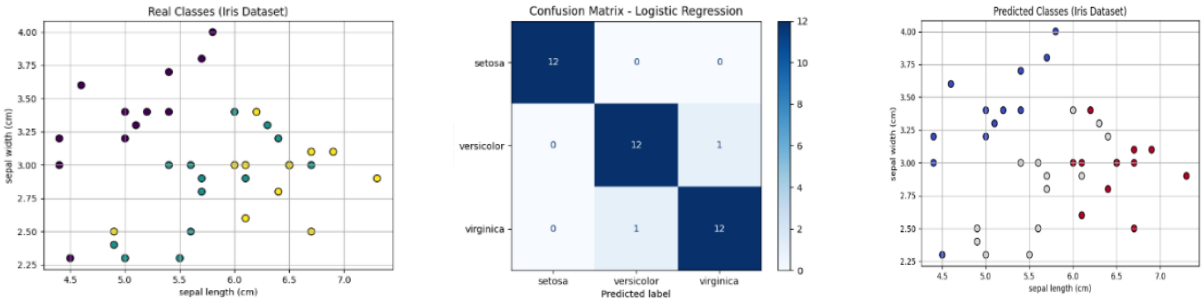
در بخش رسم نمودار، برای اینکه نشان دهیم مدل چگونه کلاس‌ها را جدا کرده، دو نمودار scatter رسم کردیم:

اولی با رنگ‌بندی بر اساس خروجی پیش‌بینی مدل (y_pred) و دومی با رنگ‌بندی بر اساس مقادیر واقعی (y_test). این مقایسه‌ی بصری باعث می‌شود بفهمیم چقدر پیش‌بینی‌های مدل نزدیک به واقعیت هستند. انتخاب رنگ‌مایه‌های coolwarm و viridis هم برای خوانایی بیشتر و تفکیک بهتر رنگ‌ها انجام شد.

در انتها، ماتریس سردرگمی (Confusion Matrix) محاسبه شد. این ماتریس نشان می‌دهد مدل چند مورد را درست و چند مورد را اشتباه تشخیص داده است. نمایش گرافیکی این ماتریس با ConfusionMatrixDisplay کمک می‌کند اشتباهات مدل خیلی واضح دیده شوند؛ مثلاً اینکه مدل دو کلاس را بیشتر با هم اشتباه گرفته یا خیر.

در مجموع، پارت 2 پروژه نشان می‌دهد چگونه می‌توان یک مسئله طبقه‌بندی واقعی را از مرحله‌ی بارگذاری داده، پیش‌پردازش، آموزش مدل، پیش‌بینی، ارزیابی و در نهایت تحلیل بصری نتایج انجام داد. این پارت پایه‌ای‌ترین مفهوم classification را در عمل به ما نشان می‌دهد و پایه‌ای برای یادگیری روش‌های پیچیده‌تر در آینده است.

Output: Scatter plot with model predictions colored and hairy Scatter plot with actual data colored and simple Confusion Matrix



در این نمودار هر نقطه یک گل است و رنگ هر نقطه نشان می‌دهد مدل آن را در کدام کلاس قرار داده. این تصویر یک درک اولیه از رفتار مدل می‌دهد و مشخص می‌کند که آیا مدل توانسته بین کلاس‌ها مرز ایجاد کند یا نه. نمودار بعدی مانند تصویر قبلی است اما رنگ‌ها بر اساس برچسب‌های واقعی تعیین شده‌اند. با مقایسه این دو نمودار می‌توان دید که مدل در کدام بخش‌ها درست عمل کرده و در کجا اشتباه داشته است. تصویر ماتریس تعداد پیش‌بینی‌های درست و غلط مدل را در هر کلاس نشان می‌دهد. خواندن آن به ما کمک می‌کند بفهمیم مدل بیشتر در کدام کلاس‌ها اشتباه می‌کند و آیا تعادل میان دقت هر کلاس حفظ شده است یا نه.

Github Link: https://github.com/Faeze-Ahmadi/advanced-programming-assignment-3/blob/main/part2_logistic_regression/classification_advanced.py

توضیح نسخه پیشرفته پارت 2 — Classification

در نسخه پیشرفته‌ی پارت دوم، هدف این بود که مسئله‌ی طبقه‌بندی روی دیتاست «ایرس» را از حالت یک تمرین ساده‌ی دانشجویی خارج کنیم و به شکل یک پروژه‌ی واقعی و قابل اتکا پیاده‌سازی کنیم. برای این کار لازم بود ساختار کد را شی‌گرا و ماژولار کنیم، از پردازش داده‌های استاندارد استفاده کنیم، روش‌های تحلیل و ارزیابی را پیشرفته‌تر کنیم، و مهم‌تر از همه خروجی‌های دقیق و قابل‌ارائه تولید کنیم.

در ابتدای کار یک کلاس به نام IrisClassifier تعریف شد که تمام منطق طبقه‌بندی داخل آن قرار می‌گیرد. این کار به چند دلیل انجام شد: اول اینکه مدیریت کد ساده‌تر شود، دوم اینکه اجزای مختلف مثل بارگذاری داده، پیش‌پردازش، آموزش، ارزیابی، مدل‌سازی و ترسیم نمودار هرکدام تابع جداگانه داشته باشند و کد از هم‌گسیخته و درهم نشود. سوم اینکه اضافه کردن ویژگی‌های جدید مثل ذخیره مدل یا نمایش PCA بدون اینکه ساختار به‌هم بریزد به راحتی انجام شود.

در ابتدای کد، کتابخانه‌هایی که استفاده شده‌اند شامل موارد زیر بود:

- کتابخانه‌ی pandas برای مدیریت داده‌ها،
- کتابخانه‌ی matplotlib برای ترسیم نمودارها،
- کتابخانه‌ی scikit-learn برای تمام مراحل یادگیری ماشین شامل بارگذاری دیتاست، مقیاس‌بندی داده‌ها، مدل‌سازی با Logistic Regression، تبدیل‌های PCA، محاسبه ماتریس سردرگمی و...

کتابخانه‌ی joblib هم برای ذخیره‌سازی مدل استفاده شد تا مدل آموزش‌دیده در یک فایل ذخیره شود و بعداً بدون نیاز به آموزش دوباره قابل بارگذاری باشد.

در کنار این‌ها از logging استفاده شد که یکی از استانداردهای مهم در پروژه‌های جدی پایتون است. Logging کمک می‌کند همه‌ی مراحل مدل‌سازی، از جمله خطاها، هشدارها و وضعیت اجرای برنامه، در خروجی ثبت شود و اگر مشکلی در اجرای برنامه یا کیفیت مدل وجود داشته باشد قابل پیگیری باشد.

پس از تعریف کلاس، اولین متدی که نوشته شد، متد load_data بود. این تابع دیتاست معروف Iris را از اسکیکیت‌لرن بارگذاری می‌کند و آن را به یک دیتافریم تبدیل می‌کند. دلیل استفاده از pandas این است که خوانایی داده‌ها بیشتر می‌شود و کار با ستون‌ها برای پیش‌پردازش راحت‌تر خواهد بود.

بعد از بارگذاری داده، متد preprocess وظیفه‌ی استانداردسازی داده‌ها را برعهده دارد. استانداردسازی به‌این دلیل انجام می‌شود که ویژگی‌های دیتاست ایرس اندازه‌های متفاوت دارند؛ مثلاً عرض گلبرگ ممکن است اعدادی کوچک داشته باشد، ولی طول گلبرگ مقدارهای بزرگ‌تری دارد. اگر داده‌ها استاندارد نشوند، مدل لوجستیک ممکن است برای برخی ویژگی‌ها وزن بیشتری قائل شود و عملکرد کلی مدل پایین‌تر بیاید. همچنین در نسخه پیشرفته‌ی ما PCA نیز استفاده شده که آن هم برای داده‌ی استاندارد بهترین کارایی را دارد؛ بنابراین استانداردسازی یک ضرورت است.

پیش از آموزش مدل، مرحله‌ی تقسیم داده به دو بخش آموزش و تست وجود دارد. این کار برای این انجام شد که ببینیم مدل روی داده‌هایی که قبلاً ندیده چطور عمل می‌کند. اگر کل داده را آموزش می‌دادیم و با همان داده ارزیابی می‌کردیم، عملاً عملکردی غیرواقعی و فریبنده می‌دیدیم.

در نسخه پیشرفته از گزینه‌ی stratify=y هم استفاده شد تا اطمینان حاصل شود که نسبت کلاس‌ها در هر دو بخش train و test یکسان باقی بماند و مدل با کمبود نمونه از یک کلاس خاص مواجه نشود.

در مرحله‌ی بعد، متد train مدل Logistic Regression را با داده‌ی استانداردشده آموزش می‌دهد. در اینجا مقدار max_iter افزایش داده شده تا زمان کافی برای همگرایی مدل وجود داشته باشد. در پروژه‌های واقعی افزایش تعداد تکرار بدون هزینه اضافه باعث پایداری بیشتر مدل می‌شود.

پس از آموزش مدل، متد evaluate مقدار دقت یا Accuracy مدل را روی داده‌ی تست محاسبه و بازگردانی می‌کند. در این نسخه پیشرفته از Error Handling هم استفاده شده تا اگر به هر دلیلی مدل هنوز آموزش داده نشده بود، پیام خطای مشخص نمایش داده شود و برنامه به‌صورت کنترل نشده کرش نکند.

در ادامه متد predict مقدار کلاس‌ها را پیش‌بینی می‌کند و این خروجی هم وارد مراحل بصری‌سازی می‌شود. بخش مهم نسخه پیشرفته این است که برای فهم بهتر رفتار مدل از PCA استفاده شده است. چون دیتاست ایرس چهار ویژگی دارد و نمایش آن‌ها در نمودارهای دوبعدی به‌صورت مستقیم چندان واضح نیست، PCA مجموعه داده را به دو مؤلفه‌ی اصلی تبدیل می‌کند که بیشترین واریانس را دارند و باعث می‌شود داده‌ها با کمترین از دست رفتن اطلاعات، در یک فضای دوبعدی قابل نمایش باشند. این کار در پروژه‌های واقعی بسیار رایج است و یکی از مهم‌ترین ابزارهای بصری‌سازی داده و مدل در یادگیری ماشین محسوب می‌شود.

در نسخه پیشرفته، دو نمودار PCA رسم شد:

- یکی بر اساس خروجی مدل (یعنی کلاس‌های پیش‌بینی‌شده)،
- و دیگری بر اساس کلاس‌های واقعی.

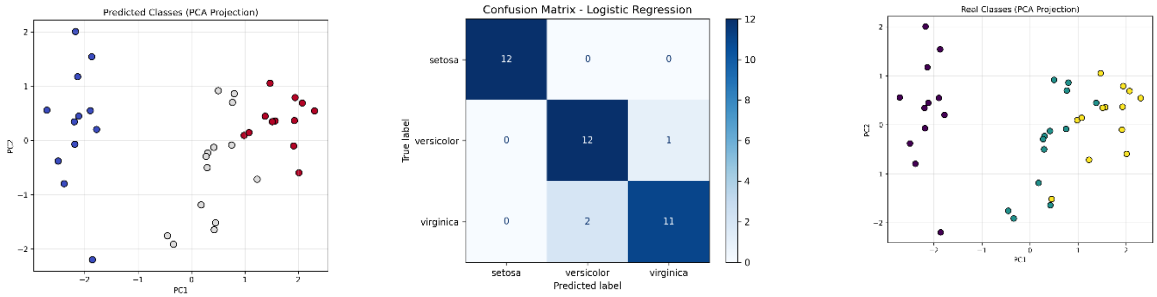
این دو نمودار مقایسه‌ای به‌وضوح نشان می‌دهد که مدل در کدام بخش‌ها خوب عمل کرده و کجا احتمال خطا یا هم‌پوشانی میان کلاس‌ها وجود دارد.

گام مهم دیگر رسم confusion matrix پیشرفته بود. ماتریس سردرگمی یکی از ابزارهای اصلی در تحلیل مدل‌های طبقه‌بندی است. در نسخه پیشرفته از نسخه گرافیکی و زیباتری استفاده شد که باعث می‌شود نتایج برای یک گزارش رسمی واضح‌تر و قابل تفسیرتر باشد.

در انتها مدل آموزش‌دیده با استفاده از joblib ذخیره شد. این عمل باعث می‌شود اگر بخواهیم نرم‌افزار یا وب‌سروری با این مدل راه‌اندازی کنیم، دیگر لازم نباشد مدل را از ابتدا آموزش دهیم و فقط با بارگذاری فایل ذخیره‌شده مدل آماده‌ی استفاده باشد.

به‌طور کلی نسخه‌ی پیشرفته‌ی پارت دوم تلاش می‌کند تمام استانداردهایی را که یک پروژه‌ی واقعی یادگیری ماشین باید داشته باشد، رعایت کند: کدنویسی ساختارمند، مدیریت خطا، لاگ‌گیری، قابل‌فهم بودن خروجی‌ها، تحلیل بصری، ذخیره‌سازی مدل و استفاده از روش‌های استاندارد پیش‌پردازش.

Output: PCA plot – model prediction and actual labels and advanced Confusion Matrix



در این نمودار داده‌ها با کمک PCA به دو بُعد فشرده شده‌اند تا بتوان ساختار درونی‌شان را مشاهده کرد. رنگ هر نقطه کلاس پیش‌بینی‌شده توسط مدل است. این تصویر کمک می‌کند الگوی کلی رفتار مدل در فضای ویژگی‌ها بهتر دیده شود. ساختار واقعی دسته‌بندی گل‌ها در فضای کاهش بعد یافته نمایش داده شده است. مقایسه این نمودار با نسخه‌ی پیش‌بینی‌شده نشان می‌دهد مدل در کدام نواحی کلاس‌ها را اشتباه گرفته یا هم‌پوشانی وجود دارد. عکس بعدی ماتریس است، این تصویر نسخه‌ی خواناتر و زیباتر ماتریس سردرگمی است. رنگ‌های گرادیانی کمک می‌کنند نواحی درست و اشتباه به‌وضوح دیده شوند و برای گزارش رسمی بسیار مناسب است.

Github Link: https://github.com/Faeze-Ahmadi/advanced-programming-assignment-3/blob/main/part3_clustering/clustering.py

توضیح نسخه ساده پارت 3 (KMeans Clustering)

در این بخش از پروژه، هدف ما این است که بدون داشتن هیچ برچسب یا اطلاعاتی درباره این‌که هر نمونه متعلق به چه کلاسی است، داده‌های مجموعه Iris را به شکل گروه‌هایی منطقی دسته‌بندی کنیم. این کار به کمک الگوریتم KMeans انجام می‌شود، که یکی از معروف‌ترین روش‌های خوشه‌بندی بدون نظارت است. نسخه ساده پارت 3 دقیقا برای این ساخته شده که مفهوم اصلی خوشه‌بندی را بدون پیچیدگی اضافی نشان دهد و مقدمات بخش پیشرفته را فراهم کند.

در ابتدای فایل، کتابخانه‌های مورد نیاز وارد می‌شوند. matplotlib.pyplot برای رسم نمودارها استفاده می‌شود. pandas برای ساخت DataFrame و کار راحت‌تر با داده. از sklearn.datasets دیتاست Iris را بارگذاری می‌کنیم. الگوریتم KMeans نیز از sklearn.cluster وارد می‌شود. و در نهایت StandardScaler برای نرمال‌سازی داده استفاده می‌شود، چون خوشه‌بندی نسبت به بزرگی یا کوچکی مقیاس‌ها حساس است.

در اولین مرحله، دیتاست Iris بارگذاری می‌شود. این مجموعه داده شامل چهار ویژگی برای هر گل است (طول و عرض گلبرگ و کاسبرگ). همچنین سه نوع گل مختلف در این دیتاست وجود دارد، اما در این نسخه، ما کاری با برچسب‌ها نداریم و فقط خود ویژگی‌ها را استفاده می‌کنیم. داده‌ها داخل یک DataFrame از نوع pandas قرار می‌گیرند تا ساختار تمیز و قابل فهمی داشته باشند. سپس ویژگی‌ها به حالت آرایه‌ای (X = df.values) استخراج می‌شوند تا برای مدل خوشه‌بندی آماده باشند.

بعد از آماده شدن داده‌ها، مرحله بسیار مهم «اسکیل کردن» یا همان نرمال‌سازی انجام می‌شود. الگوریتم KMeans فاصله نقاط را محاسبه می‌کند و اگر یک ویژگی مقدارهای خیلی بزرگ‌تری نسبت به دیگری داشته باشد، الگوریتم را منحرف می‌کند. بنابراین از StandardScaler استفاده می‌کنیم تا همه ویژگی‌ها به یک مقیاس یکسان تبدیل شوند. خروجی این مرحله X_scaled است که نسخه نرمال‌شده داده‌هاست.

در مرحله بعد، مدل KMeans ساخته و اجرا می‌شود. ما تعداد خوشه‌ها را سه قرار داده‌ایم، چون می‌دانیم دیتاست Iris در واقع سه کلاس دارد. اما حتی اگر ندانیم، هدف این مرحله این است که مدل را وادار کنیم سه گروه منطقی بسازد. با دستور fit_predict مدل آموزش می‌بیند و برای هر نمونه تعیین می‌کند متعلق به کدام خوشه است. نتیجه این فرآیند یک آرایه از عددهای 0 تا 2 است که شماره خوشه هر نمونه را مشخص می‌کند.

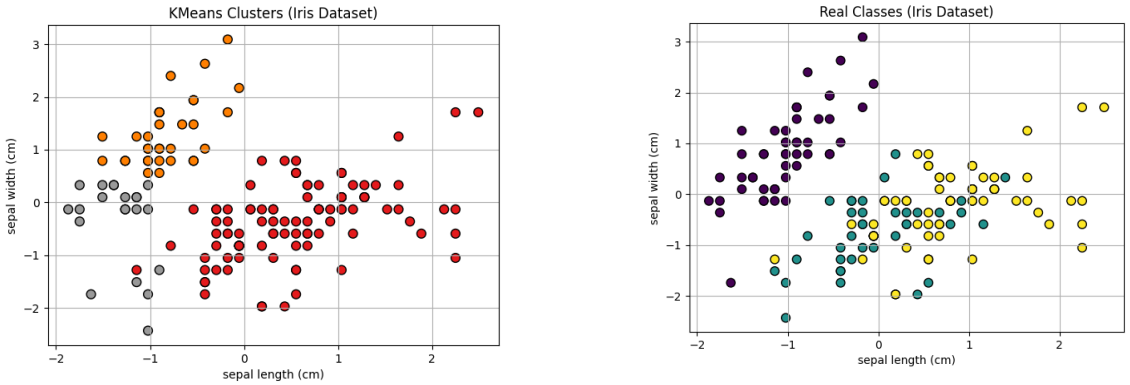
برای بهتر درک کردن نتیجه خوشه‌بندی، یک نمودار دوبعدی رسم می‌شود. فقط دو ویژگی اول (طول و عرض کاسبرگ) برای محورهای x و y استفاده می‌شوند. در نسخه ساده هیچ تبدیل پیچیده‌ای روی داده انجام نمی‌دهیم، بلکه تنها داده نرمال‌شده را نشان می‌دهیم. رنگ هر نقطه برابر با خوشه‌ای است که مدل تعیین کرده. هدف این نمودار این است که تصویری از گروه‌بندی خودکار مدل ارائه دهد.

در نمودار دوم، همین نقاط با استفاده از «برچسب واقعی» رنگ می‌شوند. یعنی رنگ نقطه‌ها نشان می‌دهد واقعاً هر نمونه متعلق به چه نوع گلی بوده. این نمودار به ما کمک می‌کند بفهمیم مدل KMeans تا چه اندازه توانسته ساختار طبیعی داده را تشخیص بدهد. هرچند این مدل «بدون نظارت» بوده، اما در اکثر موارد گروه‌هایی تشکیل می‌دهد که تا حد خوبی با کلاس‌های واقعی هماهنگ هستند.

در پایان نیز مراکز خوشه‌ها چاپ می‌شود. مرکز هر خوشه میانگین نمونه‌هایی است که در آن قرار گرفته‌اند. این مراکز در فضای ویژگی نرمال‌شده نمایش داده می‌شوند و نشان می‌دهند که «خصوصیات متوسط» هر دسته چه بوده. این اطلاعات بعدها، برای مثال در نسخه پیشرفته برای تحلیل عمیق‌تر استفاده می‌شود.

این نسخه ساده پارت 3 در مجموع یک مقدمه مناسب برای خوشه‌بندی است: بارگذاری داده، نرمال‌سازی، اجرای KMeans، رسم نتایج، مقایسه ساده با برچسب واقعی و چاپ مرکز خوشه‌ها. این بخش پایه‌ای‌ترین شکل خوشه‌بندی را نشان می‌دهد و زمینهٔ لازم را برای نسخه پیشرفته فراهم می‌کند که در آن از ابزارهایی مانند PCA، Elbow Method و Silhouette Score استفاده کردیم.

Output: KMeans and Real Labels Chart



نمودار اول نشان می‌دهد الگوریتم KMeans چگونه داده‌های استانداردشده را به سه خوشه تقسیم کرده است. رنگ هر نقطه نشان‌دهنده‌ی شماره خوشه است. هدف این تصویر نمایش خروجی خام خوشه‌بندی بدون هیچ تبدیل اضافه است. نمودار دوم همان نقاط را با رنگ‌بندی واقعی کلاس‌ها نشان می‌دهد. مقایسه آن با خروجی KMeans نشان می‌دهد الگوریتم بدون نظارت چقدر توانسته ساختار طبیعی داده را حدس بزند.

Github Link: https://github.com/Faeze-Ahmadi/advanced-programming-assignment-3/blob/main/part3_clustering/clustering_advanced.py

توضیح نسخه پیشرفته پارت 3– Clustering

در بخش سوم پروژه، سراغ یکی از مهم‌ترین مفاهیم یادگیری ماشین می‌رویم: خوشه‌بندی یا Clustering. تفاوت مهم این بخش با دو بخش قبلی این است که در اینجا برخلاف رگرسیون و طبقه‌بندی، ما هیچ برچسب یا خروجی مشخصی در اختیار نداریم. داده خام است و تنها چیزی که می‌دانیم ویژگی‌های آن است. وظیفه مدل این است که خودش الگوها را پیدا کند و داده‌ها را بر اساس شباهت‌هایی که تشخیص می‌دهد گروه‌بندی کند. این همان کاری است که الگوریتم KMeans انجام می‌دهد.

نسخه پیشرفته پارت 3 طوری طراحی شده که نه فقط خوشه‌بندی را انجام دهد، بلکه مثل یک پروژه واقعی صنعتی، شامل تحلیل، ارزیابی و مصورسازی‌های حرفه‌ای باشد. برای رسیدن به این سطح، چند مرحله مهم در کد پیاده‌سازی شده است.

در ابتدا داده‌های مجموعه معروف Iris بارگذاری می‌شود. این دیتاست شامل چهار ویژگی عددی از گل‌هاست که برای الگوریتم‌های بدون نظارت مناسب است. پس از بارگذاری داده، قدم بعدی استانداردسازی داده‌هاست. دلیل این کار این است که ویژگی‌ها دامنه‌های متفاوت دارند و اگر همین‌طور خام به الگوریتم داده شوند، ویژگی‌هایی که مقدار بزرگ‌تری دارند بر محاسبات فاصله تأثیر بیشتری می‌گذارند و نتیجه را منحرف می‌کنند. به همین دلیل از کلاس StandardScaler استفاده شد تا همه ویژگی‌ها به یک مقیاس مشترک تبدیل شوند. خروجی این مرحله داده مقیاس‌بندی‌شده‌ای است که برای خوشه‌بندی مناسب‌تر است.

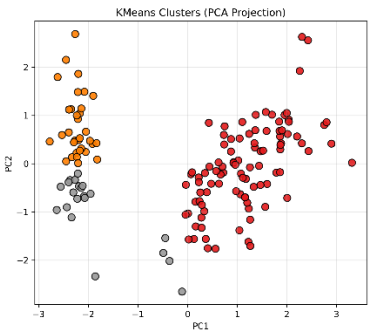
بعد از آماده‌سازی داده، یکی از مهم‌ترین مراحل این پارت انجام شد: روش Elbow. این روش کمک می‌کند تعداد بهینه خوشه‌ها انتخاب شود. چون در خوشه‌بندی نمی‌دانیم بهترین تعداد گروه‌ها چندتا باید باشد، الگوریتم را با kهای مختلف اجرا می‌کنیم و مقدار خطای درون‌خوشه‌ای (Inertia) را اندازه می‌گیریم. سپس نموداری از تغییرات این مقدار بر اساس تعداد خوشه کشیده می‌شود. جایی که نمودار «خم» می‌شود، معمولا بهترین مقدار k است. خروجی این کار یک تصویر واضح است که در پروژه ذخیره می‌شود و برای گزارش بسیار ارزشمند است.

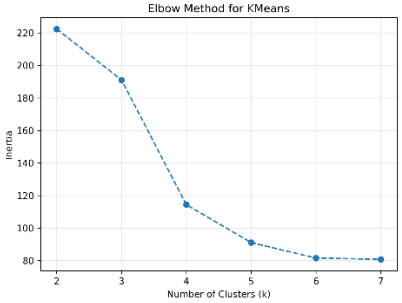
پس از مشخص کردن تعداد خوشه‌ها (که در دیتاست Iris معمولا عدد 3 انتخاب می‌شود)، مدل KMeans آموزش داده می‌شود. نسخه پیشرفته علاوه بر خود خوشه‌بندی، یک معیار بسیار مهم به نام Silhouette Score را نیز محاسبه می‌کند. این معیار کیفیت خوشه‌بندی را می‌سنجد و نشان می‌دهد که داده‌ها چقدر خوب از خوشه‌های دیگر جدا شده‌اند و چقدر با اعضای خوشه خودشان شباهت دارند. این بخش معمولاً در پروژه‌های دانشجویی دیده نمی‌شود، اما برای ارزیابی واقعی مدل ضروری است. پس از آموزش مدل، خوشه‌بندی انجام می‌شود و خروجی در قالب لیبل‌های خوشه ذخیره می‌گردد. برای اینکه نتیجه به شکلی قابل تحلیل دیده شود، داده‌ها با استفاده از PCA به دو بُعد کاهش پیدا می‌کنند و سپس خوشه‌ها روی صفحه رسم می‌شوند. PCA کمک می‌کند ساختار داده چندبُعدی را در دو بُعد قابل نمایش کنیم. در نسخه پیشرفته پروژه، دو تصویر مجزا ساخته می‌شود: تصویر نخست خوشه‌های KMeans را نشان می‌دهد، و تصویر دوم کلاس‌های واقعی دیتاست Iris را. این مقایسه یکی از جذاب‌ترین بخش‌های پروژه است و نشان می‌دهد الگوریتم بدون نظارت چقدر توانسته الگوهای واقعی را کشف کند. در مرحله بعد مرکز خوشه‌ها نمایش داده می‌شود. این اطلاعات نشان می‌دهد میانگین هر خوشه در فضای ویژگی‌های مقیاس‌بندی‌شده کجاست و می‌تواند برای تحلیل دقیق‌تر ساختار داده مفید باشد.

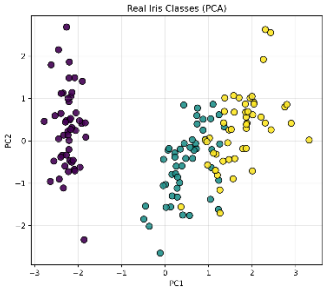
نکته مهم دیگر در نسخه پیشرفته این پارت، ذخیره‌سازی مدل است. مدل KMeans آموزش‌دیده در قالب یک فایل با پسوند pkl ذخیره می‌شود تا در آینده بتوان بدون نیاز به آموزش مجدد از آن استفاده کرد. این دقیقاً همان کاری است که در پروژه‌های واقعی انجام می‌شود.

در نهایت، تمامی نمودارها با وضوح بالا ذخیره و آماده ارائه شدند: نمودار خوشه‌بندی، نمودار کلاس‌های واقعی، PCA، و نمودار Elbow. همچنین سیستم logging در تمام مراحل رخدادها را ثبت می‌کند تا اجرای برنامه قابل پیگیری باشد. این سطح از مستندسازی و مدیریت رفتار برنامه، نسخه پیشرفته پروژه را از یک تمرین ساده دانشگاهی فراتر می‌برد و به سطح یک پروژه واقعی نزدیک می‌کند.

Output: Elbow Method and Clusters PCA and Real Labels PCA







نمودار اول تغییرات مقدار Inertia را در برابر تعداد خوشه‌ها نشان می‌دهد. نقطه‌ای که نمودار «خم» می‌شود معمولاً بهترین تعداد خوشه است. این ابزار برای انتخاب k بهینه ضروری است. در نمودار دوم داده‌ها ابتدا با PCA به دو بُعد کاهش یافته‌اند و سپس خوشه‌بندی روی آن نمایش داده شده است. این تصویر ساختار هندسی خوشه‌ها را بسیار واضح‌تر و قابل درک‌تر می‌کند و تفاوت‌های بین خوشه‌ها را برجسته می‌سازد. چارت نهایی هم داده‌های واقعی را در فضای PCA نمایش می‌دهد. مقایسه آن با نمودار خوشه‌های KMeans مشخص می‌کند که دسته‌بندی واقعی چقدر با خوشه‌بندی خودکار هم‌خوانی دارد.

به منظور دستیابی به اطلاعات بیشتر و آگاهی از روند و اجرای پروژه، به لینک گیت هاب مراجعه کنید.

با تشکر از همراهی شما.