

الله أكبر الحمد لله



University of Tehran

School of Electrical and Computer Engineering



Machine Learning

Instructor: Dr. Tavasoli-pour

Dr. Abolghasemi

Amir Hossein Shekholeslami

810101201

Tahoura Morovati

810100469

MohammadAli HashemZadeh

810100503

Spring 2023

فهرست

۶	تمیز کردن داده و استخراج ویژگی:
۱۰	چالش هزینه محاسباتی بالا
۱۱	نرمال سازی ویژگی ها
۱۱	استخراج ویژگی:
۱۲	استخراج ویژگی از نام فایل:
۱۲	استخراج ویژگی از تعداد پیکسل های موجود
۱۴	استخراج ویژگی از لبه های تصویر
۱۵	استخراج ویژگی از هیستوگرام
۱۶	استخراج ویژگی از کلیدواژه های موجود در نام تصاویر
۱۶	نتیجه گیری
۱۷	ویژگی های آماده
۱۷	طبقه بندی داده ها برحسب ویژگی های داده شده
۱۷	SVM
۱۷	اندکی درباره SVM
۱۹	اندکی درباره ماتریس درهم ریختگی
۲۲	استفاده از روش های کاهش بعد
۲۲	اندکی درباره PCA
۲۳	اندکی درباره LDA
۲۴	لاجستیک رگرشن
۲۴	اندکی درباره لاجستیک رگرشن
۲۵	Logistic Regression with PCA
۲۶	Gaussian Naïve Bayes
۲۶	اندکی درباره ی مدل Bayes
۲۷	Gaussian Naïve Bayes with PCA
۲۸	Multilayer Perceptron Neuron
۲۹	اندکی درباره MLP
۳۰	MLP with PCA
۳۱	Decision Tree
۳۱	اندکی درباره Decision Tree

۳۲Decision Tree with PCA
۳۳خوشه بندی داده‌ها برحسب ویژگی‌های استخراج شده
۳۳K-means
۳۳ K-means اندکی درباره
۳۷ GMM
۳۷ GMM اندکی درباره
۳۹ طبقه‌بندی داده‌ها برحسب ویژگی‌های استخراج شده
۳۹SVM
۴۰ SVM with PCA
۴۱ Logistic Regression
۴۲ Logistic Regression with PCA
۴۳ MLP
۴۴ Decision Tree
۴۵Decision Tree with PCA
۴۶خوشه‌بندی داده‌ها برحسب ویژگی‌های داده شده
۴۶K-means
 47
GMM 48

لینک کدها

کدها در لینک زیر یافت می‌شود:

<https://colab.research.google.com/drive/1d5B1SjQXW5KYl0yhg8RRPnesGhrOn41u?usp=sharing>

https://drive.google.com/file/d/1FCGdPzwPiHvhI9Do9ZUBe_Zj_uD1ScFY/view?usp=sharing

<https://colab.research.google.com/drive/1GPSyQnzQkmp85FKPeZ-oBfAwTTE71vVc?usp=sharing>

تمیز کردن داده و استخراج ویژگی:

در بخش اول این گزارش از ما خواسته شده است تا کارهای پیش‌پردازش داده را انجام دهیم. پیش‌پردازش داده به صورت کلی به مجموعه کارهایی گفته می‌شود که بروی داده‌های خام انجام می‌دهیم تا برای ورود به مدل آماده تر باشند. برای مثال اگر داده‌های خامی که داریم نویزی باشند، عملیاتی بروی آن انجام می‌دهیم که بتوانیم نویز را کم‌تر کنیم. یا اگر حجم داده‌ها به گونه‌ای است که هزینه‌ی محاسباتی را افزایش می‌دهد تلاش می‌کنیم تا ابعاد داده را به گونه‌ای کم کنیم که از اطلاعات مفید نیز کاسته نشود.

در این مورد در صورت پروژه، مطرح شده است تا اسامی عکس‌ها را به فرمتی مشخص کنند که در آن اطلاعاتی نظیر واقعی یا مصنوعی بودن عکس، کلاس عکس (برای مثال اینکه عکس حاوی کوه، دریا یا جنگل است) و شماره عکس است؛ همچنین در صورتی که عکس مصنوعی باشد و توسط یکی از سایت‌های نام‌برده در صورت پروژه تولید شده باشد، مشخص شده است که از کدام سایت است.

برای اینکه بتوان از این داده‌هایی که اسم هر عکس به ما می‌دهد استفاده کرد، نیاز به پیش‌پردازش داده‌ها داریم. همان‌طور که در شکل ۱ مشخص است کلاس‌های عکس‌ها بعضاً با اسامی مختلفی نوشته شده است یا اسم وبسایتی که از آن تصاویر گرفته شده است متفاوت از فرمتی است که برای آن تعریف کرده ایم، پس به این منظور شروع به پیش‌پردازش داده‌ها می‌کنیم.

در قدم اول تابعی تعریف می‌کنیم که برچسب‌های تصاویر را بگیرد و مطابق با استاندارد‌های تعریف شده، آن‌ها را تغییر دهد. چند خط از کد در پایین ضمیمه شده است. در توضیح آن می‌توان گفت که در ابتدا داده‌ها را مشاهده و خطاهای ممکن را بررسی کردیم. برای مثال در بخشی از داده‌ها به جای جدا کردن کلیدواژه Sea و

ترتیب داده با استفاده از _ ، این دو مقدار بدون هیچ فاصله ای کنار هم قرار گرفته اند که باعث می شود در آموزش یا تست داده ها این داده با این برچسب داده قابل استفاده ای محسوب نشود.

همچنین برخی از حروف به اشتباه با حروف بزرگ نوشته شده اند که با استفاده از تابع Lower آن ها را به حروف کوچک استاندارد تبدیل کردیم و در موارد که _ با - اشتباه گرفته شده است آن ها را تصحیح کردیم.

ضمناً برای اصلاح این ناهنجاری ها از کد زیر استفاده شد. که ابتدا کلمات نامناسب با کلمات مناسب جایگزین شد و در ادامه حروف آن کوچک گردید و در انتها بر روی - که جدا کننده واحدهای مختلف تشکیل دهنده نام بود، عملیات تکه بندی انجام شد.

```
splited_path = (path.replace("delle", "dalle").replace("dreamai", "dream")
    .replace("-", "_").replace("sea5", "sea_5").replace("dallebot", "dalle")
    .replace(".", "").replace("forest", "jungle").replace("see", "sea")
    .replace("dall", "dalle").replace("dallee", "dalle")
    .replace("dallem minibot", "dalle").replace("dallem ini", "dalle")
    .replace("dallem inbot", "dalle") .replace("junlge", "jungle")
    .lower().split("_"))
```

شکل ۱ - مواردی از اشتباهات موجود در اسامی فایل ها و نحوه حذف کردنشان

در اضافه تعداد برچسب ها را نیز بررسی کردیم، برای مثال اگر دانشجویی یکی از کلاس های ممکن را اضافه نکرده باشد یا به اشتباه در اسم عکس ننوشته باشد ، اسم عکس نمایش داده می شود تا عکس را پیدا کرده و تصحیح کنیم.

```

▶ from_source = ['stable', 'dalle', 'dreamstudio', 'midjourney',
                 'dream', 'bing', 'craiyon', 'none']
from_class = {'mountain': 0, 'sea': 1, 'jungle': 2}
def get_feature_from_path(path):

    splited_path = (path.replace("delle", "dalle").replace("dreamai", "dream")
                    .replace("-", "_").replace("sea5", "sea_5").replace("dallebot", "dalle")
                    .replace(".", "").replace("forest", "jungle").replace("see", "sea")
                    .replace("dall", "dalle").replace("dallee", "dalle")
                    .replace("dallem minibot", "dalle").replace("dallem ini", "dalle")
                    .replace("dallem in bot", "dalle").replace("junlge", "jungle")
                    .lower().split("_"))

    if (len(splited_path) != 5):
        print(splited_path)
    source = splited_path[2]
    image_class = splited_path[3]
    if source not in from_source:
        print(source, path)

    if image_class not in from_class:
        print(image_class, path)

    return source, from_class[image_class]

```

شکل ۲ - تابعی که توسط عملیات پیش پردازش داده‌ها را انجام می‌دهیم.

□

تصاویر داده شده دارای فرمت‌های متنوعی هستند. بنابراین لازم است ابتدا این فرمت‌ها شناسایی شود و برای استخراج ویژگی اقدامات لازم صورت گیرد. بدین منظور ابتدا فرض بر آن گذاشته شد که تصاویر از دو فرمت png، jpg ایجاد شده‌اند. اما برای آنکه تصاویر با سایر فرمت‌ها از دست نرود، نام فایل‌هایی که از فرمت‌های فوق نبودند نمایش داده شد تا بدین طریق بتوان فرمت‌های آنها را نیز پشتیبانی کرد. بنابراین دو فرمت jpeg و jfif نیز به فرمت‌های تصاویر اضافه شدند. لازم به توضیح است یکی از تصاویر که از نوع واقعی نیز بود فرمت webp داشت که از گذاشتن این تصویر در مجموعه داده نهایی خودداری شد.

```

labels = list()
for filename in range(len(df_label)):
    if (df_label['810101213_fake_stable_mountain_2.png'][filename].endswith(".jpeg") or df_label['810101213_fake_stable_mountain_2.png'][filename].endswith(".jpg") or
        df_label['810101213_fake_stable_mountain_2.png'][filename].endswith(".png") or df_label['810101213_fake_stable_mountain_2.png'][filename].endswith(".jfif") or
        df_label['810101213_fake_stable_mountain_2.png'][filename].endswith(".webp")):

```

شکل ۳ - چگونگی اضافه کردن فرمت‌های مختلف تصویر

در آخر نیز، سائز برخی تصاویر به قدری بالا بود که در هنگام لود آنها برنامه دچار مشکل می شد، بنابراین با استفاده از بلوک مدیریت خطا این موارد مدیریت شد و در نهایت تعداد دو عدد از تصاویر واقعی که دچار این مشکل می شدند در مجموعه داده نهایی گذاشته نشدند.

```
try:
    labels.append(get_labels((df_label['810101213_fake_stable_mountain_2.png'][filename])))
except Exception as e:
    print(e, filename)
else:
    print(filename)
    print(df_label['810101213_fake_stable_mountain_2.png'][filename])
```

شکل ۴ - چگونگی حذف کردن داده هایی با حجم بالا

• تقسیم داده ها

در مرحله بعدی باید طبق دستورکار داده ها را به داده های آموزش و تست تقسیم بندی کنیم. درصد تخصیص داده ها به تست و آموزش می تواند بر روی نتیجه خروجی تاثیر گذار باشد.

داده ها در مدل های یادگیری ماشین به آموزش، تست و اعتبارسنجی تقسیم می شوند تا عملکرد مدل های در داده های جدید بررسی شود و از بروز بیش برآزش جلوگیری شود.

تقسیم بندی در داده ها به طور کلی به صورت زیر انجام می شود:

۱. آموزش: این مجموعه داده برای آموزش مدل استفاده می شود. مدل روی این داده ها پارامترهای خود را بهبود می بخشد.

۲. تست: این مجموعه داده برای ارزیابی عملکرد مدل استفاده می شود. مدل بر روی این داده ها تست می شود و عملکرد آن بررسی می شود.

۳. اعتبارسنجی: این مجموعه داده برای تنظیم پارامترهای مدل استفاده می‌شود. مدل بر روی این داده‌ها تنظیم می‌شوند و پارامترهای بهتری برای مدل انتخاب می‌شوند.

نرخ مناسب برای تقسیم بندی داده‌ها به طور کلی به مسئله و حجم داده‌ها بستگی دارد. به طور معمول، می‌توان ۷۰-۸۰ درصد از داده‌ها را برای آموزش، ۱۰-۱۵ درصد از داده‌ها را برای اعتبارسنجی و ۱۰-۱۵ درصد برای تست استفاده کرد. لازم به ذکر است که این نرخ قابل تغییر است و باید با توجه به مسئله و داده‌های خاص تعیین شود.

در این مسئله در ابتدا مقدار اولیه ۲۰ درصد را برای داده‌های تست تعیین می‌کنیم، بدیهی است که ۸۰ درصد داده‌ها برای آموزش مورد استفاده قرار می‌گیرند.

```
0s X_train, X_test, y_train, y_test = train_test_split(np.array(df), np.array(labels), test_size=0.2, random_state=42)
```

شکل ۵ - تابع تقسیم داده به داده‌های آموزش و تست

برای تقسیم داده‌ها از تابع آماده بالا استفاده می‌کنیم. در این تابع پارامتری وجود دارد که نیاز به توضیح بیشتر دارد و آن `random_state` می‌باشد.

این مقدار برای توضیح مقدار تصادفی بودن داده‌هاست. به این معنی که این اطمینان را حاصل کنیم که در هر بار توزیع داده‌ها، آن‌ها به همان روشی تولید می‌شوند که در بارهای قبل شده‌اند.

زمانی که یک مقدار به خصوص را برای `random_State` مشخص می‌کنید داده‌ها به صورت یکسان تقسیم می‌شوند بدون توجه به این که چندبار تقسیم شوند. این مهم زمانی ارزش پیدا می‌کند که می‌خواهیم عملکرد مدل خود را با مدل‌های دیگر مقایسه کنیم. لازم به ذکر است که مقداری که به این پارامتر می‌دهیم مادامی که در دوره‌های مختلف یکسان باشد تفاوتی در نتیجه حاصل نخواهد کرد.

چالش هزینه محاسباتی بالا

به طور کلی تصاویر داری اندازه‌های متفاوتی بودند. بنابراین باید اندازه تصاویر یکسان سازی می‌شد. این سائز باید به گونه‌ای انتخاب می‌شد که اولاً اطلاعات کلی تصاویر از دست نرود، دوماً حجم دیتا خیلی زیاد نباشد، چرا که به طور کلی محدودیت منابع پردازشی وجود دارد و باید این موارد مدیریت شود. سائزهای متفاوتی برای اینکار در نظر گرفته شد، مثلاً با ابعاد ۳۲ در ۳۲ تست‌های اولیه گرفته شد که دقتی حدود ۶۴ درصد بر روی مدل SVM بدست آمد. با دو برابر کردن ابعاد دقت حدود ۴ درصد افزایش می‌یابد تا جایی که در ابعاد ۱۰۰ در ۱۰۰ این دقت به حدود ۷۲ درصد بر روی داده‌های تست می‌رسید. اما همانطور که گفته شد ابعاد بالاتر باعث می‌شدند اولاً پردازش‌ها کند شود و هم اینکه در برخی موارد RAM پر شود. بنابراین سائز نهایی تصاویر برابر ۱۰۰ در نظر گرفته شد.

نرمال سازی ویژگی ها

از آنجا که ویژگی‌ها از انواع مختلف هستند و به طور مثال برخی از ویژگی‌ها مقدار یک پیکسل در تصویر بوده و برخی نیز صرفاً یک شناسه هستند، باید نرمال سازی روی آنها انجام شود تا تمامی ویژگی‌ها به صورت یکسان روی فرایند یادگیری تاثیر بگذارند. البته در تست‌های مختلف دیده شد در برخی موارد عدم نرمال سازی باعث دقت بهتر بر روی داده‌های تست می‌شود که می‌تواند به این علت باشد که به نوعی با عدم انجام اینکار ویژگی‌هایی که همه متعلق به یکی از دسته‌های بالا بودند، به دلیل عددهای هم بازه در کنار هم معنا پیدا کنند و وزن‌های نسبتاً شبیه به هم داشته باشند و برای ویژگی‌های یک دسته دیگر این عدد متفاوت باشد.

```
scaler = MinMaxScaler()  
X_train = scaler.fit_transform(X_train)  
X_test = scaler.transform(X_test)
```

نکته دیگر اینکه در فرایند آماده سازی Scaler باید از داده‌های آموزش استفاده شود و صرفاً اعمال تغییرات روی داده‌های تست صورت گیرد، زیرا در طول فرایند آموزش و آماده سازی ما اطلاعی از داده‌های تست نداریم.

استخراج ویژگی:

ویژگی های استخراج شده در این لینک یافت می‌شود:

https://drive.google.com/file/d/1ZoZ0rCrKemzNS_2IEm6nqWmO-kJgp9KX/view?usp=sharing

لیبل ها ویژگی های استخراج شده

<https://drive.google.com/file/d/1-fwLttwMEFi0Fz99q-yftrpNpCzuEepn/view?usp=sharing>

ویژگی های کاهش یافته توسط PCA

<https://drive.google.com/file/d/1C1PsvwQeYtbGdIRhMITNEyzmAuNvTRfx/view?usp=sharing>

لیبل های کاهش یافته توسط PCA

https://drive.google.com/file/d/1-7b1L-7FqaMXFijxXowPxx4xfxSrYF_h/view?usp=sharing

به طور کلی برای استخراج ویژگی از جنبه های مختلف به تصویر نگاه شده است. در واقع در دو دسته از ویژگی ها تصاویر از فیلتر عبور داده شده اند و مقدار هر پیکسل تصویر بعد از عبور از فیلتر به عنوان یک ویژگی در نظر گرفته شده است. فیلترها باعث می شوند مواردی که مد نظر است بیشتر به چشم بیاید. مثلاً یکی از فیلترها از جنبه استخراج حاشیه ها و دیگری از جنبه میزان روشنایی کلی تصویر توزیع و میانگین رنگ تصویر را آماده می کند. در زیر دسته های ویژگی مورد استفاده بررسی شده اند.

استخراج ویژگی از نام فایل:

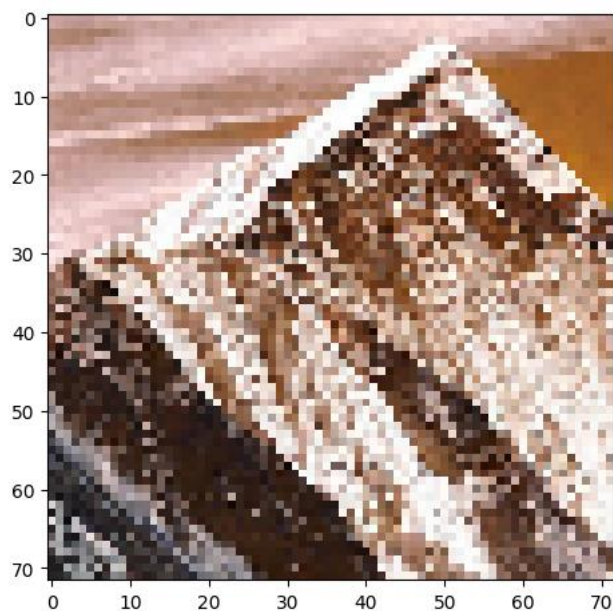
در نام فایل ها اطلاعات ارزش مندی نظیر دسته بندی تصویر، منبع تصویر وجود داشت. در مورد منبع تصویر باید **none** توجه داشت این ویژگی به تنهایی می تواند داده ها را به صورت خطی جدا کند، تصاویری که در منبع قرار می گیرند، تصاویر واقعی هستند و بقیه منابع برای تصاویر غیر واقعی بوده است. ولی از آنجا که به طور کلی این ویژگی باید جزو ویژگی های نامشخص باشد از آن استفاده نشد. ولی اینکه این تصاویر جزو کدام دسته هستند (جنگل، دریا یا کوه) به عنوان یک ویژگی برای استفاده در مدل در نظر گرفته شد. نکته مهم اینکه این دسته بندی باید به یک شاخص عددی تبدیل می شد که برای کوه عدد ۰، برای دریا عدد ۱ و برای جنگل عدد ۲ در نظر گرفته شد.

استخراج ویژگی از تعداد پیکسل های موجود

اولین دسته ویژگی که به تعداد پیکسل‌های تصویر، ویژگی می‌دهد، در واقع میانگین رنگ‌های سبز و قرمز و آبی را به ازای هر پیکسل محاسبه می‌کند. همانطور که گفته شد این عدد برای هر پیکسل نشان می‌دهد که در آن پیکسل چقدر رنگ‌ها اشباع هستند، زیرا اگر فرض کنیم نسبت توزیع رنگ‌ها ثابت است، با زیاد شدن هر رنگ به نسبت مشخص این میانگین اضافه شده و در واقع رنگ‌ها اشباع بیشتری دارند. از طرفی اگر فرض کنیم در تصاویر مصنوعی رنگ‌ها اشباع تر هستند می‌توان این دسته ویژگی را جزو ویژگی‌های نهایی در نظر گرفت. به طور کلی باید ذکر شود دسته ویژگی‌های زیادی بررسی شد ولی این دسته ویژگی به تنهایی می‌توانست دقت حدود ۶۰ درصد برای ما تامین کند که نشان دهنده ارزشمندی بالای آن است.

```
def get_average_of_pixels(image):
    feature_matrix = np.zeros((72, 72))
    for i in range(0, image.shape[0]):
        for j in range(0, image.shape[1]):
            feature_matrix[i][j] = ((int(image[i,j,0]) + int(image[i,j,1]) + int(image[i,j,2]))/3)
    return np.reshape(feature_matrix, 72*72)
```

همانطور که در بالا مشاهده می‌شود، در زمان عکس برداری قطعه کد ساین تصویر برابر ۷۲ در ۷۲ در نظر گرفته شده و در این تست دقتی در حدود ۷۴ درصد برای مدل SVM بدست آمده بود.



شکل ۶- نمونه‌ای از تصویر پس از کاهش حجم

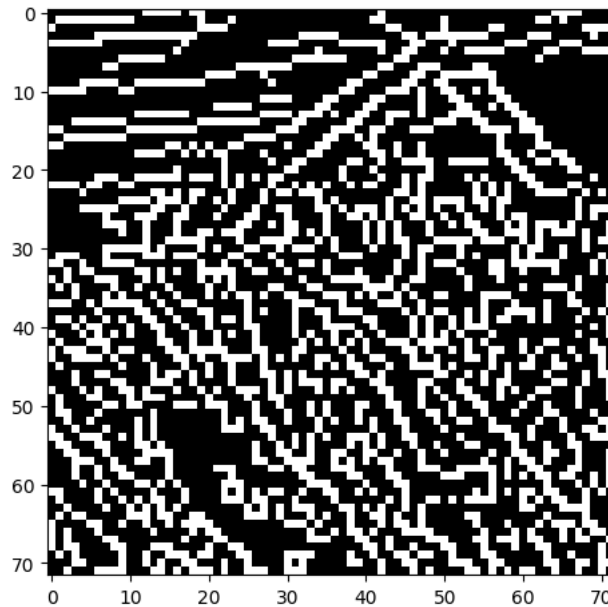
تصویر فوق نتیجه اعمال این فیلتر روی یکی از تصاویر غیر واقعی کوه است. دلیل اینکه تصویر رنگی به نظر می‌رسد استفاده از نقشه رنگی است. که در پلات اتفاق افتاده است.

استخراج ویژگی از لبه های تصویر

دومین دسته از ویژگی که مجدداً به تعداد پیکسل‌های تصویر، ویژگی می‌دهد، حاشیه‌های موجود در تصویر را نمایان می‌کند. ابتدا تصویر ورودی با استفاده از فیلتر نرمال می‌شود تا نویزها کاهش یابد و لبه‌های واقعی روشن‌تر شوند. سپس مشتقات جزئی تصویر نسبت به جهت‌های افقی و عمودی محاسبه می‌شوند تا گرادیان تصویر بدست آید. برای هر پیکسل در تصویر، مقدار و جهت گرادیان محاسبه می‌شود. مقدار گرادیان نشان دهنده قدرت تغییرات شدت رنگ در آن نقطه است و جهت گرادیان نشان دهنده جهت افقی یا عمودی لبه است. نقاطی که مقدار گرادیان آن‌ها کمتر از حد آستانه تعیین شده است، به عنوان نقاط غیر لبه حذف می‌شوند. نقاطی که مقدار گرادیان آن‌ها از حد آستانه‌ای بیشتر است، به عنوان نقاط پتانسیل لبه انتخاب می‌شوند. برای تولید این دسته ویژگی از کتابخانه cv2 و دستور Canny استفاده شده است و آستانه آن ۵۰ و ۱۵۰ در نظر گرفته شد.

```
def get_edge_of_image(image):  
    edges = cv2.Canny(image, 50, 150)  
    return np.reshape(edges, 72*72)
```

همانطور که در بالا مشاهده می‌شود، در زمان عکس برداری قطعه کد سائز تصویر برابر ۷۲ در ۷۲ در نظر گرفته شده بود. ضمناً تصویر ورودی از قبل سیاه و سفید شده است. در تست‌های مختلف برای نمایان شدن حاشیه‌ها از روش‌های دیگر مانند Laplacian نیز استفاده شد ولی نتایج بدست آمده چندان رضایت بخش نبود. که البته می‌تواند به دلیل سائز کوچک تصاویر در آن تست‌ها بوده باشد، زیرا در زمان این تست سائز تصاویر ۳۲ در ۳۲ بود.



شکل ۷ - نمونه ای تصویر پس از استخراج لبه ها

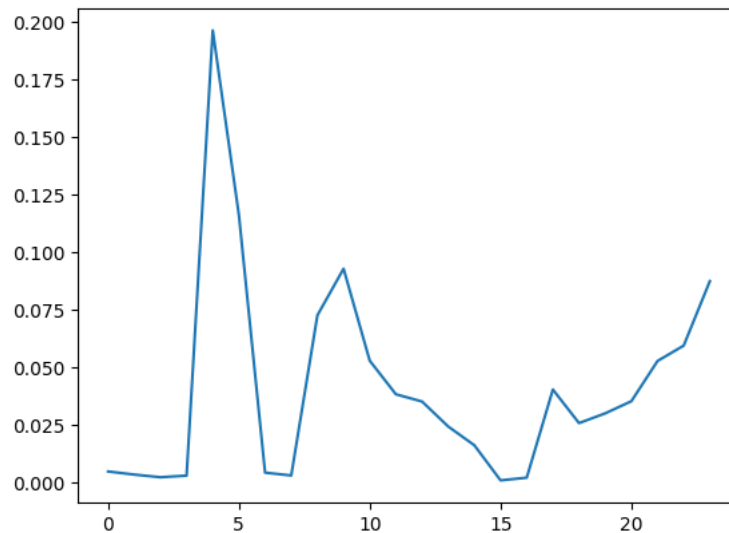
تصویر فوق حاصل اعمال این فیلتر روی تصویر کوه می باشد که به دلیل وجود تغییرات رنگی زیاد به شکل بالا درآمده است. البته در بالای تصویر خطوط کوه مشخص است.

استخراج ویژگی از هیستوگرام

سومین دسته از ویژگی مربوط به اطلاعات هیستروگرام عکس می باشد که مجموعاً ۲۴ ویژگی به ویژگی ها می افزاید. ورودی تصویر پس از HSV شدن فضای رنگی با استفاده از تابع `calcHist` هیستوگرام تصویر در هر کانال فضای رنگ HSV محاسبه می شود.

```
def get_color_histogram(image, bins=8):
    # Convert the image to the HSV color space
    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    hist_hue = cv2.calcHist([hsv_image], [0], None, [bins], [0, 180])
    hist_saturation = cv2.calcHist([hsv_image], [1], None, [bins], [0, 256])
    hist_value = cv2.calcHist([hsv_image], [2], None, [bins], [0, 256])
    hist_features = np.concatenate((hist_hue, hist_saturation, hist_value)).flatten()
    hist_features /= hist_features.sum()
    return hist_features
```

شکل ۸ - کدی که برای گرفتن میانگین هیستوگرام استفاده می کنیم



شکل ۹ - ۲۴ ویژگی استخراج شده مربوط به تصویر کوه

تصویر فوق ۲۴ ویژگی مربوط به عکس کوه را نمایش می دهد.

استخراج ویژگی از کلیدواژه های موجود در نام تصاویر

کلاس مربوط به هر تصویر که متعلق به دسته دریا یا کوه یا جنگل می باشد نیز به عنوان یک ویژگی به مدل داده می شود. همان طور که پیشتر نیز توضیح داده شد در اسم هر تصویر برجسبی وجود دارد که می توان از آن برای ویژگی هر تصویر استفاده کرد. در این از سه برجسب کوه و دریا و جنگل با مقادیر کمی استفاده شده است.

نتیجه گیری

لازم به توضیح است در این میان به روش های مختلف ویژگی های بالا اعمال شد، مثلا در برخی موارد به جا اینکه تمام پیکسل های عکس فیلتر شده به عنوان ویژگی در نظر گرفته شود، میانگین آنها در نظر گرفته شد یا در موردی دیگر جمع آنها اعمال شد ولی بهترین نتیجه در شرایطی بدست آمد که کل پیکسل ها به عنوان ویژگی در نظر گرفته می شد.

ویژگی های آماده

در این پروژه، در کنار تصاویر داده شده فایل CSV نیز تحت عنوان feature در دسترس قرار گرفته شده است..

این ویژگی ها با استفاده از شبکه عصبی efficient net استخراج شده است و بهینه ترین مجموعه ویژگی ممکن برای آموزش مدل هاست در نتیجه قبل از آموزش داده ها با ویژگی های خودمان می توانیم با تقریب خوبی تخمین بزنیم که درصد دقت کلاس بندی و تمایز داده ها زمانی که با ویژگی های داده شده باشند بالاتر از زمانی خواهد بود که با ویژگی ها خودمان باشند.

برای پیاده سازی مسیر پروژه، در ابتدا روش های خوشه بندی و طبقه بندی ذکر شده در متن گزارش را با ویژگی هایی که در اختیار دادیم آموزش داده و دقت را میسنجیم و سپس با استفاده از ویژگی های که خودمان تعریف کرده مقایسه می کنیم.

طبقه بندی داده ها برحسب ویژگی های داده شده

در این قسمت از ما خواسته شده است تا با استفاده از ویژگی های داده شده و همچنین ویژگی هایی که خودمان استخراج کردیم، طبقه بندی داده ها را انجام دهیم. در این قسمت از روش های LogisticRegression، SVM، DecisionTree، MLP، NavieBayesian استفاده کردیم.

در ادامه به شرح هر کدام می پردازیم.

SVM

اندکی درباره SVM

یک الگوریتم یادگیری ماشین برای مسائل دسته بندی و SVM (Support Vector Machines) روش رگرسیون است. این الگوریتم براساس ایده هایی از هندسه فضایی بر پایه داده ها عمل می کند

در SVM، هدف اصلی پیدا کردن یک صفحه (در حالت دو بعدی) یا یک فضای تصمیم (در حالت چند بعدی) است که بین دو دسته از داده ها قرار می گیرد واصله کمینه را با داده های هر دسته داشته باشد. این صفحه یا فضا به عنوان "صفحه جداکننده" شناخته می شود.

در SVM، نقاطی که نزدیکترین فاصله را به صفحه جاکنده دارند و به عنوان "بردارهای پشتیبان" مشخص می‌شوند. این بردارهای پشتیبان هستند زیرا تعیین کننده موقعیت و شکل صفحه جاکنده هستند.

روش SVM برای دسته‌بندی داده‌های خطی و غیرخطی قابل استفاده است. در حالت خطی، SVM از تابع هسته (kernel function) استفاده می‌کند تا داده‌ها را به یک فضای بالاتر منتقل کند و در فضا، یک صفحه خطی جاکنده را پیدا کند. در حالت غیرخطی، SVM از توابع هسته غیرخطی مانند تابع گاوسی (Gaussian) استفاده می‌کند تا داده‌ها را به یک فضای دیگر برده و در این فضا، یک صفحه غیرخطی جاکنده را پیدا کند.

روش SVM دارای مزایا و محدودیت‌های خود است. مزیت اصلی آن قابلیت استفاده در مسائل با ابعاد و داده‌های نامتعادل است. همچنین، SVM قابلیت کنترل برزش بیش از حد (overfitting) را دارد. اما، محدودیت اصلی آن نیاز به تنظیم پارامترهای مهم مانند پارامتر C و انتخاب تابع مناسب است.

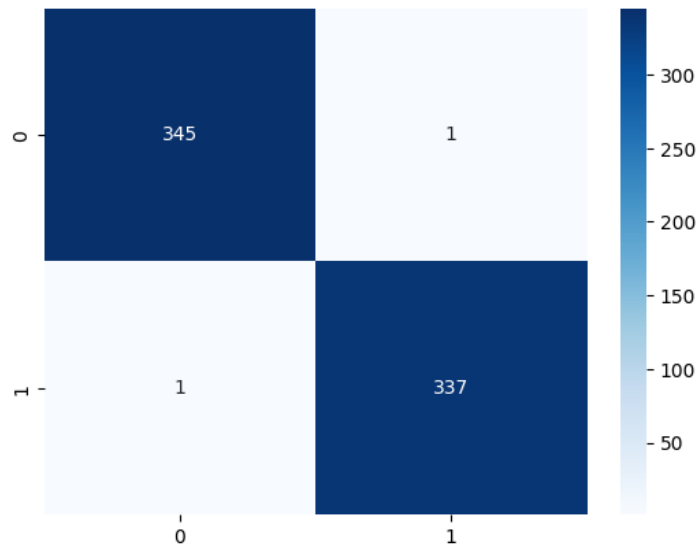
در کل، SVM یک روش قوی و محبوب در یادگیری ماشین است.

در ابتدا با استفاده از ویژگی‌های داده شده SVM را پیاده‌سازی کرده و به دقت زیر رسیدیم:

accrracy of svm is 99.71					
	precision	recall	f1-score	support	
0	1.00	1.00	1.00	346	
1	1.00	1.00	1.00	338	
accuracy			1.00	684	
macro avg	1.00	1.00	1.00	684	
weighted avg	1.00	1.00	1.00	684	

شکل ۱۰ - پارامترهای ارزیابی مدل SVM برای ویژگی‌های آماده

همچنین ماتریس پراکندگی آن به صورت زیر است:



شکل ۱۱ - ماتریس درهم ریختگی برای مدل SVM

اندکی درباره ماتریس درهم ریختگی

فرم کلی ماتریس درهم ریختگی به صورت زیر است :

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

شکل ۱۲ - فرمت کلی ماتریس درهم ریختگی

وقتی داده ها را بدست می آوریم ، بعد از تمیز کردن داده ها و پیش پردازش ، اولین قدمی که انجام می دهیم ، مدل سازی داده ها و استفاده از مدل طراحی شده است. اما چگونه می توانیم اثربخشی مدل خود را اندازه

بگیریم. اثربخشی بهتر ، عملکرد بهتر و دقیقاً همان چیزی است که ما می خواهیم. و اینجاست که ماتریس درهم ریختگی (Confusion Matrix) مورد توجه قرار می گیرد. این ماتریس می تواند عملکرد سیستم یادگیری ماشین را بررسی کند. این ماتریس یک روش اندازه گیری عملکرد برای مساله طبقه بندی یادگیری ماشین است که در آن خروجی می تواند دو یا چند کلاس باشد. این جدول با ۴ ترکیب مختلف از مقادیر پیش بینی شده و واقعی ساخته می شود.

ماتریس درهم ریختگی برای محاسبه معیار های ارزیابی دقت (Accuracy)، صحت (Precision)، فراخوانی (Recall) و F1 لازم و ضروری است.

با یک مثال از موضوع بارداری این مفاهیم را توضیح می دهیم.

۱. مثبت صحیح : (True Positives) پیش بینی کردیم که مثبت بوده و درست پیش بینی کردیم

- پیش بینی کردیم خانم باردار است و درست بوده است

۲. منفی صحیح : (True Negatives) پیش بینی کردیم منفی باشد و درست پیش بینی کردیم.

- پیش بینی کردیم آقا باردار نباشد و درست بوده است.

۳. مثبت کاذب : (False Positives) پیش بینی کردیم مثبت باشد ولی غلط پیش بینی کردیم

- پیش بینی کردیم آقا باردار باشد! ولی غلط بود.

۴. منفی کاذب : (False Negatives) پیش بینی کردیم غلط باشد ولی درست بود.

- پیش بینی کردیم خانم باردار نباشد ولی واقعا باردار بود.

با استفاده از این ۴ مفهوم، معیار های مختلفی برای ارزیابی مدل یادگیری ماشین تعریف می شود.

در ادامه معروف ترین این معیار ها را بررسی می کنیم.

فراخوانی (Recall)

به این مفهوم اشاره دارد که از بین همه کلاسهای مثبت ، چقدر درست پیش بینی کردیم. باید تا حد ممکن بالا باشد.

$$Recall = \frac{TP}{TP + FN}$$

شکل ۱۳ - ریکال

صحت: (Precision)

یعنی از بین تمام کلاسهای مثبتی که به طور صحیح پیش بینی کرده ایم ، چند نفر در واقع مثبت هستند.

$$Precision = \frac{TP}{TP + FP}$$

شکل ۱۴ - پریسیژن

معیار F۱

اگر بخواهیم همزمان هر دو معیار صحت و فراخوانی در ارزیابی مدل دخیل باشند از این معیار استفاده می کنیم.

$$F - measure = \frac{2 * Recall * Precision}{Recall + Precision}$$

شکل ۱۵ - معیار F۱

پس با استفاده از اطلاعات بدست آمده درباره ماتریس درهم ریختگی یا پراکندگی می توان گفت که در مورد SVM مقدار داده هایی که به اشتباه مثبت تشخیص داده شده اند صفر و مقدار داده هایی که با اشتباه منفی تشخیص داده شده اند ۱ عدد بوده است و تعداد داده هایی که به درستی تشخیص داده شده اند ۳۴۶+۳۳۷ می باشد. که این مقادیر با درصد دقت که ضمیمه شده نیز تطابق دارد.

اندکی درباره PCA

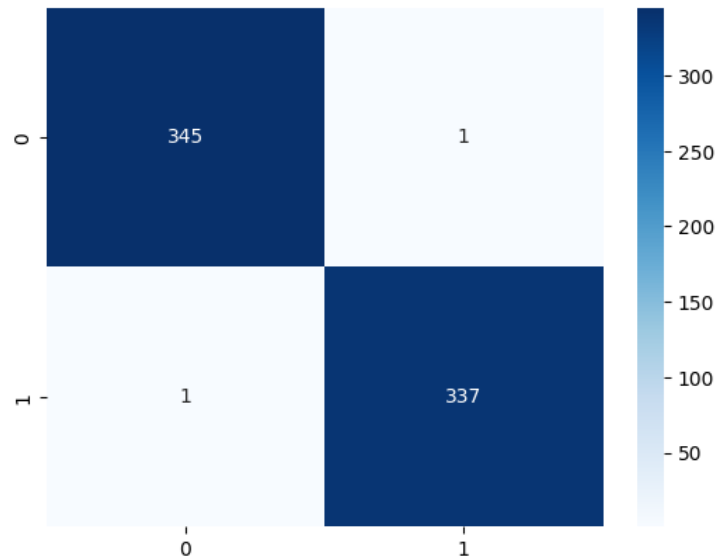
گاه‌ها ممکن است در کاربردهای مختلف با حالت‌هایی مواجه می‌شویم که در آن‌ها ابعاد دیتاست بسیار بزرگ باشد به عبارت دیگر تعداد فیچرها بسیار زیاد است. این بزرگ بودن سبب می‌گردد که پیچیدگی داده‌ها و در نتیجه پیچیده‌تر شدن مدل‌های ما می‌شود، که این امر باعث افزایش حجم محاسبات می‌گردد. هم چنین قابلیت تفسیرپذیری ما نیز کاهش می‌یابد. لذا با توجه به تمامی موارد گفته شده نیاز به روش‌هایی جهت کاهش ابعاد داریم.

از آنجا که دو تصویر فیلتر شده به همراه چند ویژگی دیگر را به عنوان ویژگی نهایی در نظر گرفته شده، ابعاد ویژگی بسیار زیاد بوده (در حد چند هزار ویژگی) و باید ابعاد نهایی با استفاده از روش‌های کاهش بعد کم شود. به همین خاطر از دو روش PCA و LDA استفاده شد که در ادامه در مورد معایب و مزایای هر روش صحبت می‌کنیم.

در این روش که به صورت **unsupervised** انجام می‌شود، محور مختصات به گونه‌ای انتخاب می‌شود که داده‌ها پس از آنکه بر روی آن **project** شدند، بیشتری واریانس را داشته باشند. به عبارت دیگر بیشترین اطلاعات ممکن از آن تعداد از ابعاد داده که نیاز است را نگه می‌دارد. در این روش ابعاد نهایی ویژگی مقادیر ۵۰ ۱۰۰ ۲۰۰ ۵۰۰ ۱۰۰۰ تست شد که بهترین دقت روی ۵۰۰ بعد بدست آمد. در دو شکل زیر نتایج حاصل از اعمال PCA بر روی داده‌ها را مشاهده می‌کنیم که دقت و مقادیر داخل ماتریس پراکندگی تغییری نکرده است.

accrracy of SVM is 99.71				
	precision	recall	f1-score	support
0	1.00	1.00	1.00	346
1	1.00	1.00	1.00	338
accuracy			1.00	684
macro avg	1.00	1.00	1.00	684
weighted avg	1.00	1.00	1.00	684

شکل ۱۶ - پارامترهای ارزیابی پس از اعمال PCA در مدل SVM



شکل ۱۷ - پارامترهای ارزیابی پس از اعمال PCA در مدل SVM

اندکی درباره LDA

در این روش کاهش بعد به صورت supervised انجام می‌شود، و از آنجا که تعداد ابعاد بسیار زیاد بوده و احتمالاً این تعداد داده با این تعداد ویژگی به صورت خطی کاملاً جدایی پذیر هستند، به نظر می‌رسد این روش از دست رفتن بیشتر Generalization مدل شده و ارزیابی روی داده‌های تست را با نتایج خیلی بهتری رو به رو نمی‌کند. به عبارت دیگر دقت با روش کاهش بعد PCA بالا بود و در نهایت روش اول انتخاب گردید.

ذکر این نکته نیز خالی از لطف نیست که اگر از روش‌های کاهش بعد استفاده نمی‌کردیم، علاوه بر اینکه فرایند آموزش به شدت کند می‌شد دقت نیز تا ۱۰ درصد کمتر می‌شد که از جمله دلایل آن می‌توان به بایاس شدن مدل روی داده‌های ترین اشاره کرد، چرا که با این تعداد ویژگی احتمال جدایی پذیری ۱۰۰

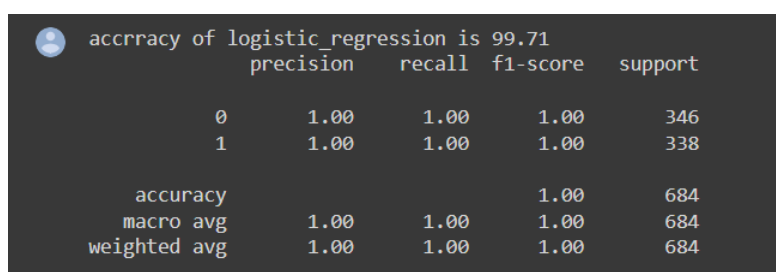
درصدی داده‌ها بسیار بالا بوده و لزوماً حاشیه‌ی اطمینان در نظر گرفته شده هم نمی‌تواند بایاس بالای مدل را که ناشی از اهمیت اشتباه به ویژگی‌های بی‌ارزش است کم کند.

لاجستیک رگرشن^۱

اندکی درباره لجستیک رگرشن

رگرسیون لجستیک (Logistic Regression) یکی از الگوریتم‌های یادگیری ماشین است. این الگوریتم برای مسائل طبقه‌بندی (Classification) استفاده می‌شود که در آن متغیر وابسته‌ی گسسته (Categorical) مطرح می‌شود.

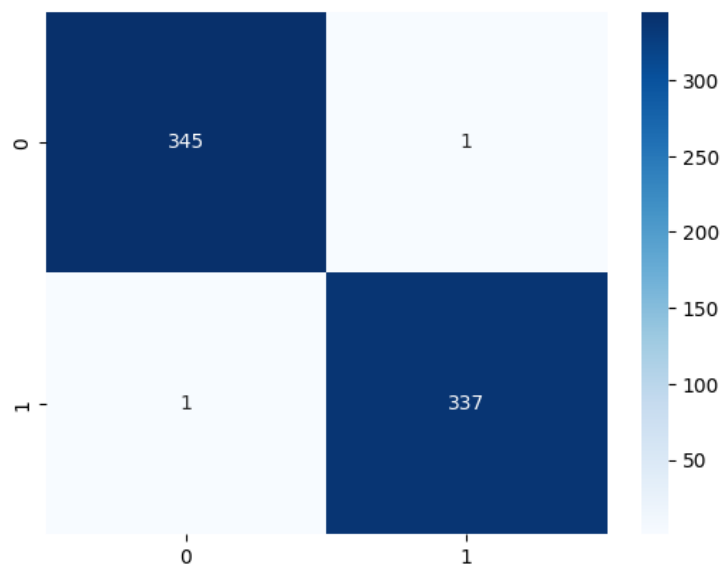
در این قسمت از ویژگی‌های آماده استفاده کرده و با استفاده از متد لجستیک رگرشن آن‌ها را آموزش داده ایم. نتایج به شرح زیر است:

A terminal window with a dark background and light blue text. It shows the output of a logistic regression model. The first line says 'accrracy of logistic regression is 99.71' (note the typo 'accrracy'). Below this is a table with five columns: 'precision', 'recall', 'f1-score', and 'support'. The first two rows show results for classes 0 and 1, all with values of 1.00 and support of 346 and 338 respectively. The last three rows show aggregated metrics: 'accuracy' (1.00, 684), 'macro avg' (1.00, 684), and 'weighted avg' (1.00, 684).

	precision	recall	f1-score	support
0	1.00	1.00	1.00	346
1	1.00	1.00	1.00	338
accuracy			1.00	684
macro avg	1.00	1.00	1.00	684
weighted avg	1.00	1.00	1.00	684

شکل ۱۸ - پارامترهای ارزیابی در مدل *logisticRegression*

¹ Logistic Regression



شکل ۱۹ - ماتریس درهم ریختگی در مدل *Logistic Regression*

Logistic Regression with PCA

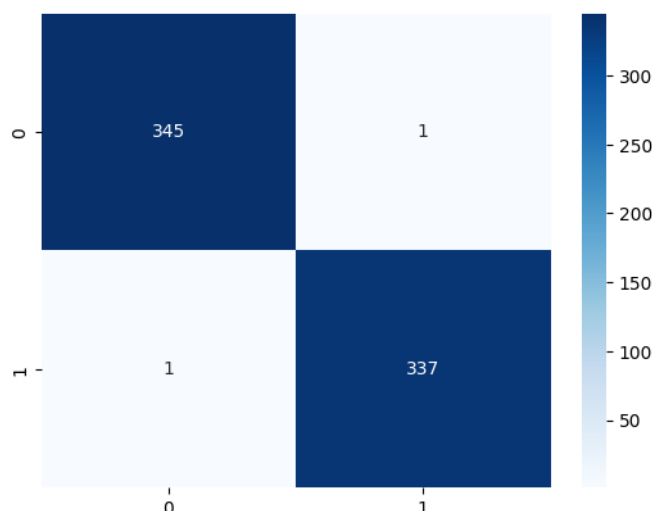
```

accuracy of logistic_regression is 99.71
precision    recall  f1-score   support

      0       1.00      1.00      1.00      346
      1       1.00      1.00      1.00      338

 accuracy          1.00      684
  macro avg       1.00      1.00      1.00      684
 weighted avg     1.00      1.00      1.00      684
  
```

شکل ۲۰ - پارامترهای ارزیابی پس از اعمال *PCA* در مدل *LogisticRegression*



شکل ۲۱ - ماتریس درهم ریختگی در مدل *Logistic Regression* پس از اعمال *PCA*

Gaussian Naïve Bayes

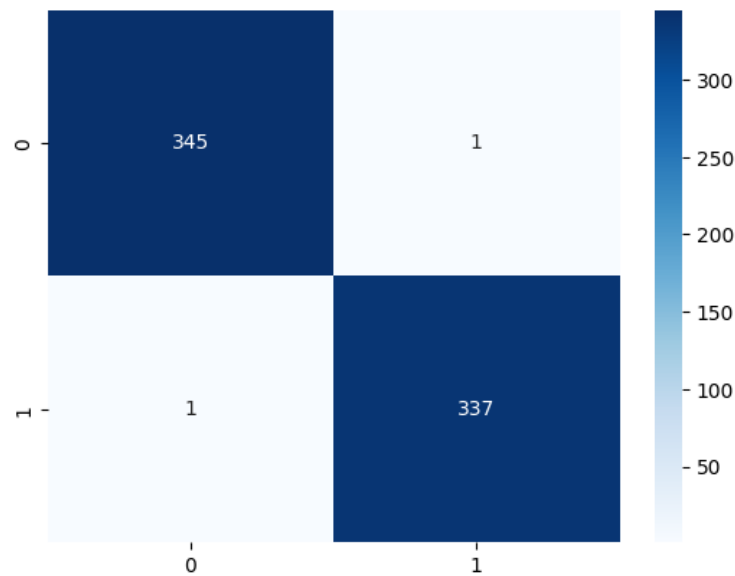
اندکی درباره ی مدل Bayes

این یک روش طبقه بندی بر اساس قضیه بیز با فرض استقلال در بین پیش بینی کننده ها است. به زبان ساده، یک طبقه بند **Naive Bayes** فرض می کند که وجود یک ویژگی خاص در یک کلاس با وجود ویژگی دیگر ارتباطی ندارد. به عنوان مثال، میوه ای اگر قرمز، گرد و حدود ۳ اینچ قطر داشته باشد، ممکن است یک سیب در نظر گرفته شود. حتی اگر این ویژگی ها به یکدیگر یا ویژگی های دیگری بستگی داشته باشند، همه این خصوصیات به طور مستقل در احتمال سیب بودن این میوه نقش دارند و به همین دلیل به “بیز ساده” معروف است. ساخت مدل دسته بندی بیز ساده **Naive Bayes** آسان است و مخصوصاً برای مجموعه داده های بسیار بزرگ مفید است. همراه با سادگی، ثابت شده که این الگوریتم حتی از روش های طبقه بندی بسیار پیچیده نیز پیشی می گیرد.

در اینجا داده ها را با استفاده از مدل مذکور آموزش داده و سپس عملکرد مدل را ارزیابی کردیم. نتایج پارامترهای ارزیابی در تصاویر زیر قابل مشاهده است.

accrracy of Gaussian is 99.71					
	precision	recall	f1-score	support	
0	1.00	1.00	1.00	346	
1	1.00	1.00	1.00	338	
accuracy			1.00	684	
macro avg	1.00	1.00	1.00	684	
weighted avg	1.00	1.00	1.00	684	

شکل ۲۲ - پارامترهای ارزیابی در مدل *Naïve Bayes*



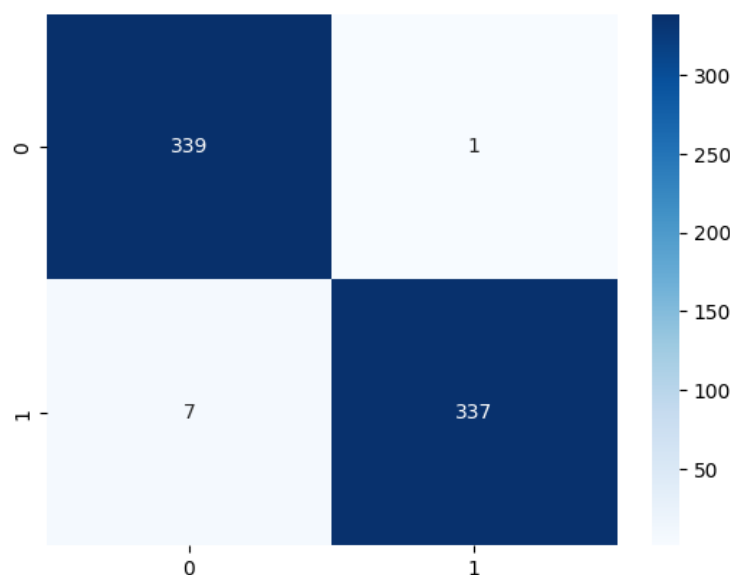
شکل ۲۳ - ماتریس درهم ریختگی در مدل *Naïve Bayes*

Gaussian Naïve Bayes with PCA

در این قسمت با استفاده از ویژگی های آماده ای که در اختیار داریم، داده ها را با مدل *Naïve Bayes* آموزش دادیم اما در این بار به جهت کاهش ابعاد داده و هزینه های محاسباتی از *PCA* استفاده می کنیم. نتایج به شرح زیر است.

accuracy of Gaussian is 98.83					
	precision	recall	f1-score	support	
0	1.00	0.98	0.99	346	
1	0.98	1.00	0.99	338	
accuracy			0.99	684	
macro avg	0.99	0.99	0.99	684	
weighted avg	0.99	0.99	0.99	684	

شکل ۲۴ - پارامترهای ارزیابی در مدل *Naïve Bayes* پس از اعمال *PCA*



شکل ۲۵ - ماتریس درهم ریختگی در مدل *Naïve Bayes* پس از اعمال *PCA*

همان طور که میبینیم در این مدل به نسبت مدل های قبل تر زمانی که *PCA* را اعمال کردیم درصد دقت پایین تر آماده است، دلیل آن را می توان در سادگی مدل *Naïve Bayes* جستجو کرد. مسئله این است که در مدل های دیگر چون پیچیدگی بیشتر است در نهایت بازهم مقداری داده اضافی خواهیم داشت اما در *Bayes* چون مدل، مدل ساده ای است پس بازهم به همه ویژگی ها احتیاج دارد و با *PCA* که بخشی از ویژگی ها را به ویژگی های دیگری تبدیل می کنیم و ابعاد را کاهش می دهیم، ابعاد حذف شده باعث می شود بخشی از اطلاعات موجود از بین برود و دقت پایین تر بیاید.

Multilayer Perceptron Neuron

MLP از یک روش یادگیری نظارت شده به نام "پس انتشار" برای آموزش شبکه استفاده می کند که برای محاسبه گرادیان تابع ضرر ، نیاز به خروجی مد نظر و معلوم برای هر مقدار ورودی دارد MLP. ، یک پرسپترون خطی استاندارد اصلاح شده است و می تواند داده هایی را که به طور خطی تفکیک پذیر نیستند از هم تفکیک کند.

در قسمت زیر با استفاده از فیچرهای آماده، مدل MLP را آموزش دادیم و نتایج آن به شرح زیر است.

```

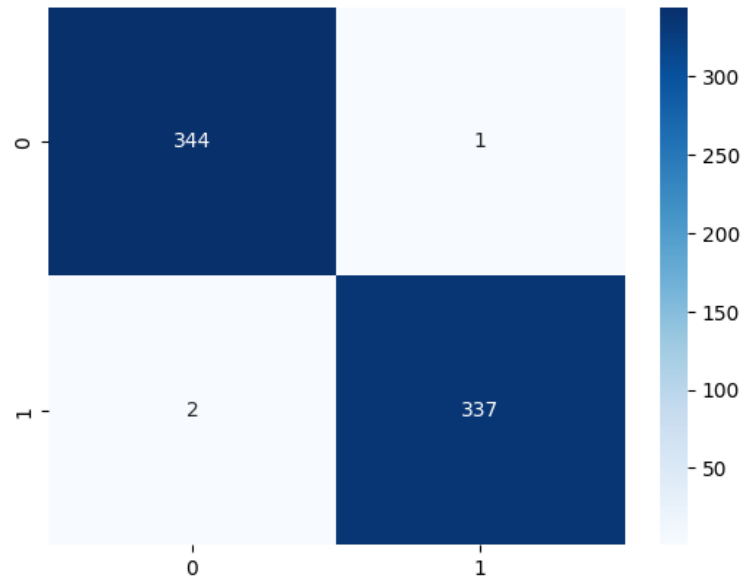
→ accrracy of Gaussian is 99.56
      precision    recall  f1-score   support

         0         1.00      0.99      1.00         346
         1         0.99      1.00      1.00         338

 accuracy
macro avg      1.00      1.00      1.00         684
weighted avg    1.00      1.00      1.00         684

```

شکل ۲۶ - پارامترهای ارزیابی در مدل MLP



شکل ۲۷ - ماتریس درهم ریختگی در مدل MLP

MLP with PCA

```

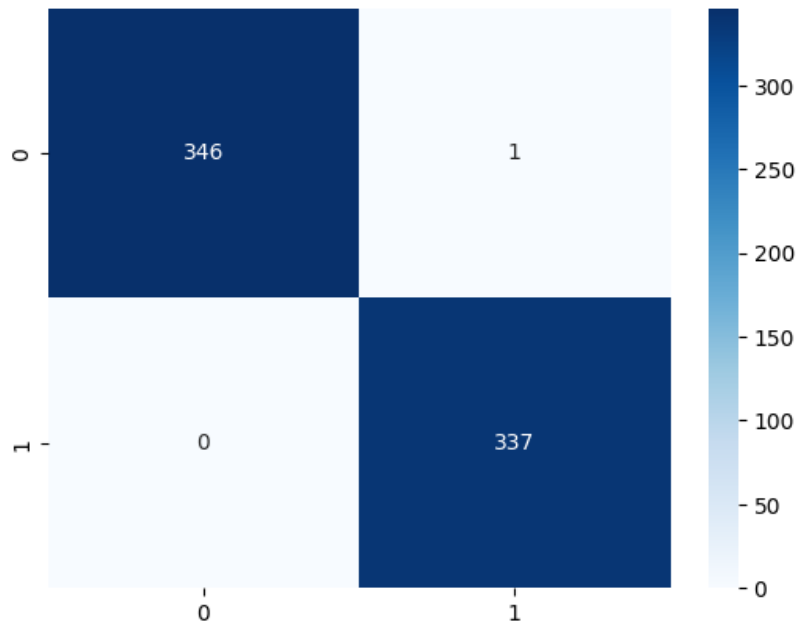
accracy of Gaussian is 99.85
      precision    recall  f1-score   support

     0       1.00      1.00      1.00       346
     1       1.00      1.00      1.00       338

 accuracy          1.00         684
 macro avg          1.00         684
 weighted avg       1.00         684

```

شکل ۲۸ - پارامترهای ارزیابی در مدل MLP پس از اعمال PCA



شکل ۲۹ - ماتریس درهم ریختگی در مدل PCA پس از اعمال MLP

Decision Tree

اندکی درباره Decision Tree

یادگیری درخت تصمیم (Decision tree learning) گروهی از الگوریتم‌های یادگیری ماشین هستند که در طبقه‌بندی آماری کاربرد دارند. درخت‌های تصمیم به گروه الگوریتم‌های یادگیری تحت نظارت تعلق دارند و بیشتر آنها بر اساس حداقل‌سازی کمیتی به نام آنتروپی ساخته می‌شوند.

```

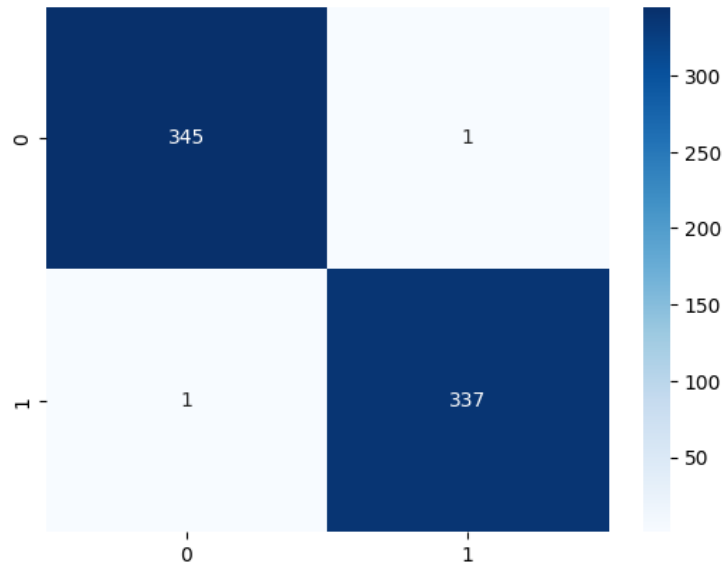
accrracy of Gaussian is 99.71
      precision    recall  f1-score   support

     0         1.00      1.00      1.00        346
     1         1.00      1.00      1.00        338

 accuracy          1.00          684
 macro avg         1.00          684
 weighted avg      1.00          684

```

شکل ۳۰ - پارامترهای ارزیابی در مدل decision Tree

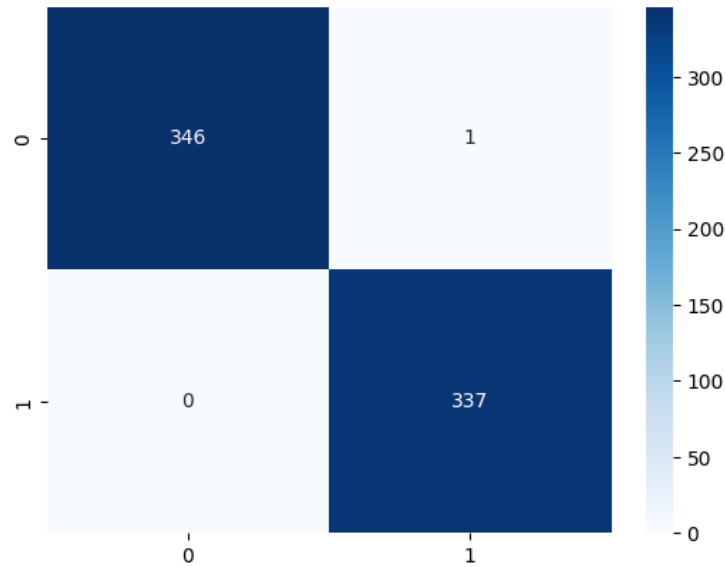


شکل ۳۱ - ماتریس درهم ریختگی در مدل *Decision Tree*

Decision Tree with PCA

accrracy of Gaussian is 99.85					
	precision	recall	f1-score	support	
0	1.00	1.00	1.00	346	
1	1.00	1.00	1.00	338	
accuracy			1.00	684	
macro avg	1.00	1.00	1.00	684	
weighted avg	1.00	1.00	1.00	684	

شکل ۳۲ - پارامترهای ارزیابی در مدل *Decision Tree* پس از اعمال *PCA*



شکل ۳۳ - ماتریس درهم ریختگی در مدل Decision Tree پس از اعمال PCA

خوشه بندی داده‌ها برحسب ویژگی‌های استخراج شده

تا به حالا دسته بندی یا طبقه بندی داده ها را بررسی کردیم از این پس به خوشه بندی داده‌ها میپردازیم. خوشه بندی یک روش Unsupervised می باشد.

K-means

اندکی درباره K-means

الگوریتم K-means یکی از روش های خوشه بندی ساده و سریع است. این الگوریتم دارای یک پارامتر به نام k است که تعداد خوشه هایی که باید به دست آید را مشخص می کند. الگوریتم K-means پایه به صورت زیر است:

K داده را به عنوان مرکز خوشه انتخاب می کنیم، سپس فواصل بقیه داده‌ها با مرکز خوشه‌ها را تعیین می کنیم و داده‌هایی که به مرکز هر خوشه نزدیک تر هستند را در آن خوشه قرار می دهیم. میانگین هر خوشه را به عنوان مرکز جدید خوشه انتخاب می کنیم این مراحل را تا زمانی ادامه می دهیم که خوشه‌ها بدون تغییر باقی بمانند. در ادامه مراحل الگوریتم K-means بیان شده است.

ورودی: خصوصیات n داده و k تعداد دسته ها

خروجی k : دسته که داده های هر دسته از نظر شباهت به هم نزدیک و از دسته های دیگر دورند.

k داده را به عنوان مرکز خوشه انتخاب می کنیم.

مرحله سوم تا پنجم را تا رسیدن به عدم تغییر در خوشه ها تکرار می کنیم.

فواصل بقیه داده ها با مرکز خوشه ها را تعیین می کنیم.

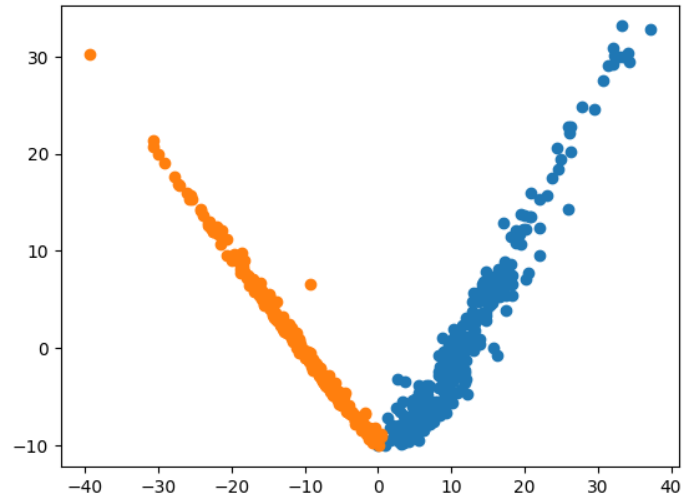
داده هایی که به مرکز هر خوشه نزدیکترند در آن خوشه قرار می گیرند.

میانگین هر خوشه را به عنوان مرکز جدید خوشه در نظر می گیریم.

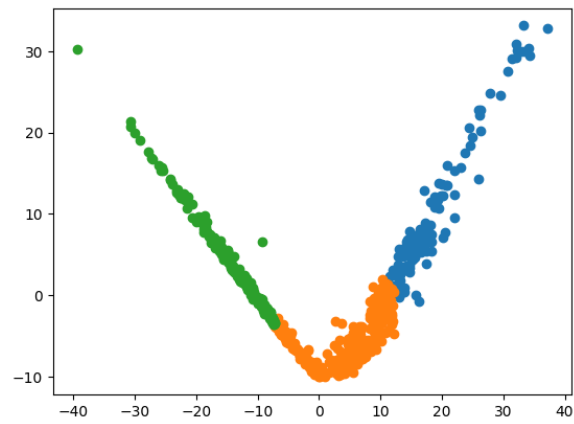
به طور معمول، مرکز خوشه های اولیه به صورت تصادفی از میان نمونه های اولیه گزینش می شوند. به همین دلیل، مرکز خوشه های اولیه در دو خوشه بندی مستقل K -means می توانند متفاوت باشند. این موضوع موجب می شود که خوشه های به جا مانده از دو اجرای مختلف K -means با هم متفاوت باشند. بنابراین همواره به بهینه ی سراسری نمی رسد اما ممکن است به بهینه ی محلی برسد. در الگوریتم K -means می توان از معیار های فاصله ی گوناگون بهره گرفت و خوبی یا بدی به کارگیری آن معیار بستگی به نوع داده هایی دارد که باید خوشه بندی شوند.

اگر یک ماتریس را به عنوان ورودی به K -means بدهیم، K -means این ماتریس را داده با خصوصیت در نظر می گیرد و اگر k را دو در نظر بگیریم داده ها را به دو دسته تقسیم می کند با این شرط که اعضای در یک دسته قرار می گیرند که خصوصیات آنها به یکدیگر نزدیک تر باشد یعنی سطر متناظر با آنها در ماتریس ورودی شباهت بیشتری به یکدیگر داشته باشد.

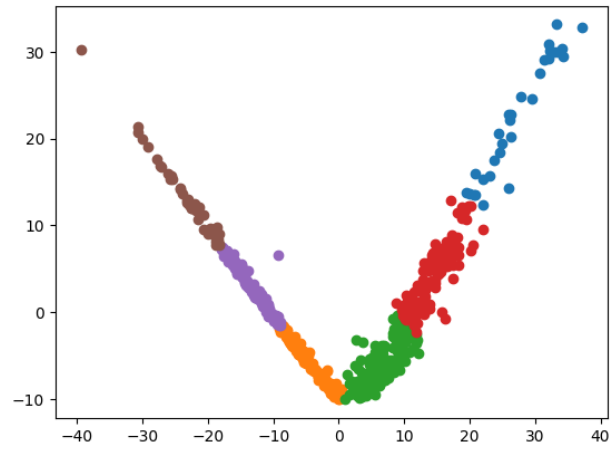
در این قسمت خروجی های حاصل را نمایش می دهیم.



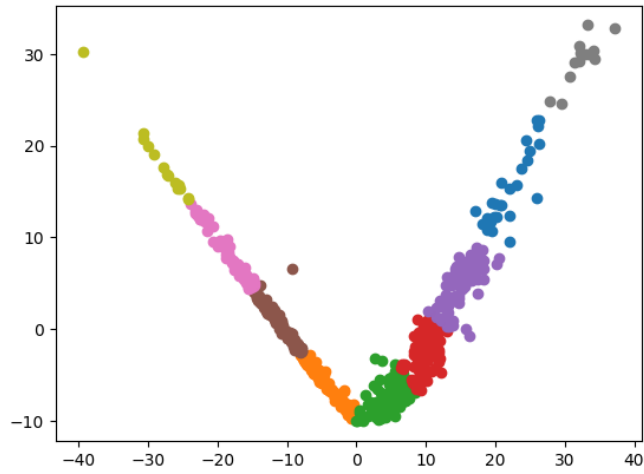
شکل ۳۴ - خروجی الگوریتم K-means با ۲ اجزا



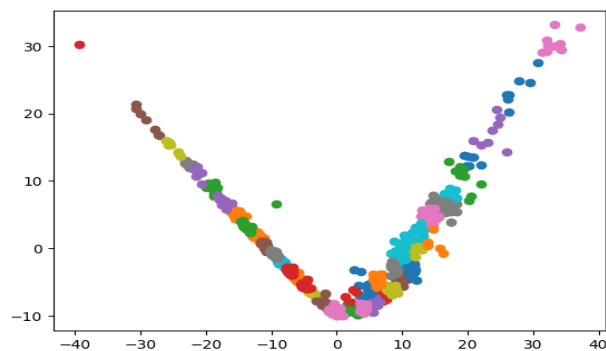
شکل ۳۶ - شکل ۳۴ - خروجی الگوریتم K-means با ۳ اجزا



شکل ۳۷ - شکل ۳۴ - خروجی الگوریتم K-means با ۶ اجزا



شکل ۳۷ - شکل ۳۴ - خروجی الگوریتم K-means با ۹ اجزا

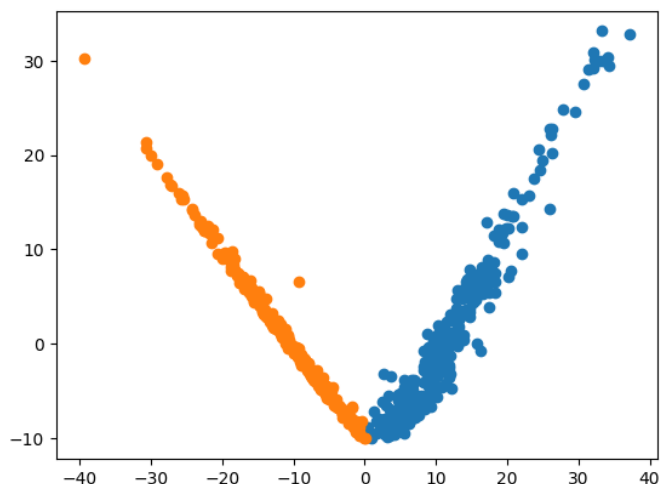


شکل ۳۸ - شکل ۳۴ - خروجی الگوریتم K-means با ۵۰ اجزا

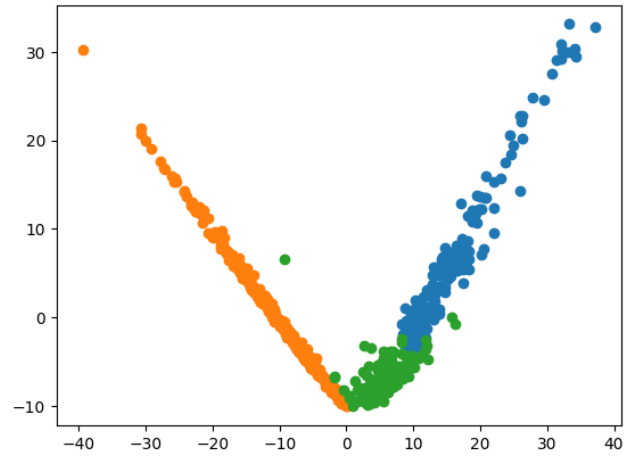
GMM

اندکی درباره GMM

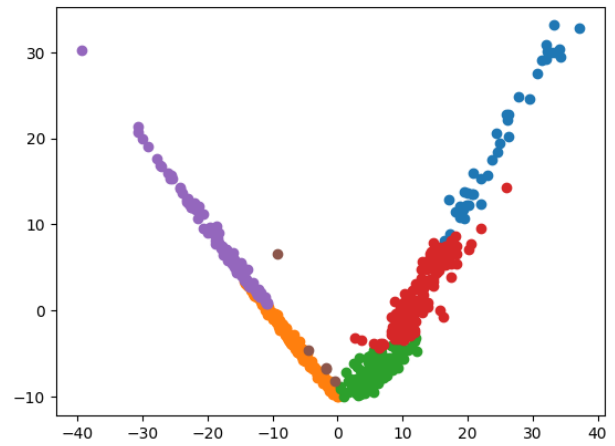
به منظور نشان دادن نماینده ای از یک زیرجمعیت توزیع شده نرمال، در کل جمعیت، ما از مدل مخلوط گوسی استفاده می کنیم GMM. به داده هایی که زیرجمعیت ها به آن تعلق دارند، نیازی ندارد. این به مدل اجازه می دهد تا بطور خودکار، زیرجمعیت ها را یاد بگیرد. از آنجا که ما از وظایف زیرجمعیت آگاهی نداریم، این امر تحت یادگیری بدون نظارت قرار می گیرد.



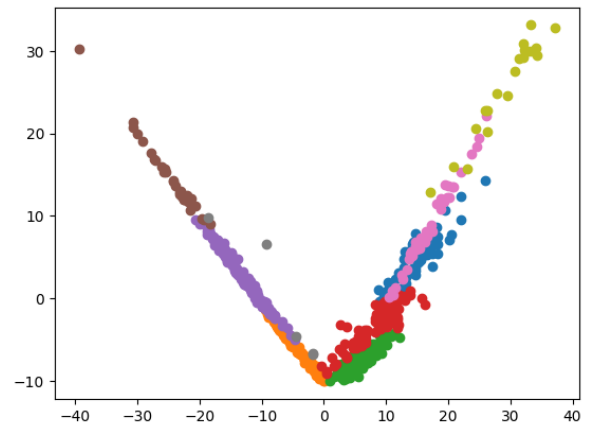
شکل ۳۵ - خروجی الگوریتم GMM با ۲ اجزا



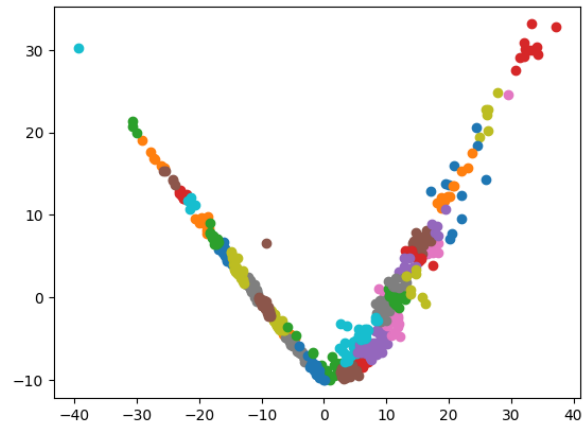
شکل ۳۶ - خروجی الگوریتم GMM با ۳ اجزا



شکل ۳۷ - خروجی الگوریتم GMM با ۴ اجزا



شکل ۳۸ - خروجی الگوریتم GMM با ۵ اجزا



شکل ۳۹ خروجی الگوریتم GMM با ۵۰ اجزا

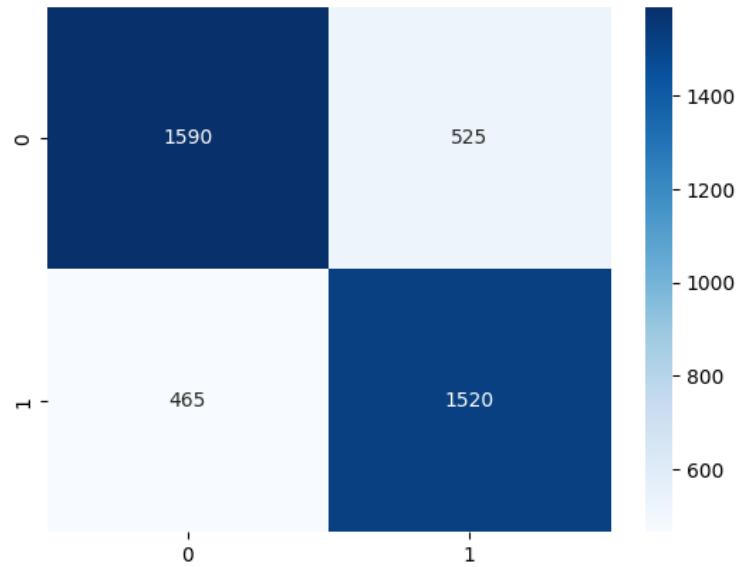
طبقه‌بندی داده‌ها برحسب ویژگی‌های استخراج شده

همان طور که در بخش های قبل درباره حجم محاسباتی بالا صحبت به میان آمد، داده‌ها را با تعداد پیکسل‌های مختلفی آموزش دادیم. به جهت بررسی تاثیر تغییر تعداد پیکسل‌ها مقادیر مختلفی را امتحان کرده و در کد آورده ایم اما در اینجا برای مفید به فایده بودن گزارش تنها نتایج گرفته شده از داده‌های ۱۰۰ در ۱۰۰ پیکسلی را نمایش می‌دهیم که به طور معمول در میان مدل‌های مختلف نتایج بهتری نیز داده است.

SVM

accrracy of SVM is 75.85					
	precision	recall	f1-score	support	
0	0.75	0.77	0.76	2055	
1	0.77	0.74	0.75	2045	
accuracy			0.76	4100	
macro avg	0.76	0.76	0.76	4100	
weighted avg	0.76	0.76	0.76	4100	

شکل ۴۰ - پارامترهای ارزیابی در مدل SVM زمانی که ابعاد پیکسل‌ها ۱۰۰ در ۱۰۰ است

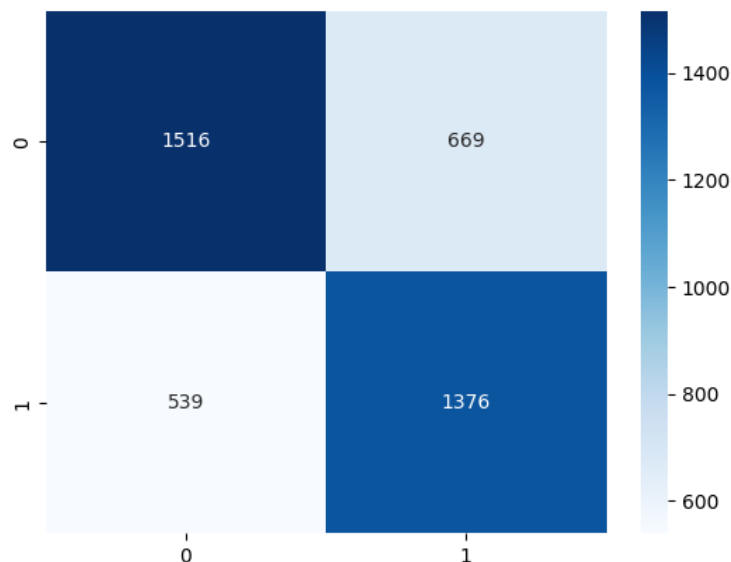


شکل ۴۱ - ماتریس درهم ریختگی در مدل *SVM* زمانی که ابعاد پیکسل ها ۱۰۰ در ۱۰۰ است

SVM with PCA

accrracy of SVM is 72.90					
	precision	recall	f1-score	support	
0	0.71	0.77	0.74	2055	
1	0.75	0.69	0.72	2045	
accuracy			0.73	4100	
macro avg	0.73	0.73	0.73	4100	
weighted avg	0.73	0.73	0.73	4100	

شکل ۴۲ - پارامترهای ارزیابی در مدل *SVM* پس از اعمال *PCA* زمانی که ابعاد پیکسل ها ۱۰۰ در ۱۰۰ است



شکل ۴۳ - ماتریس درهم ریختگی در مدل *SVM* زمانی که ابعاد پیکسل‌ها ۱۰۰ در ۱۰۰ است.

همان طور که مشخص است که به دلیل اینکه ابعاد بسیار زیادی کم شده است، دقت کاهش می یابد اما هزینه محاسباتی به طور قابل توجهی کاهش می یابد.

Logistic Regression

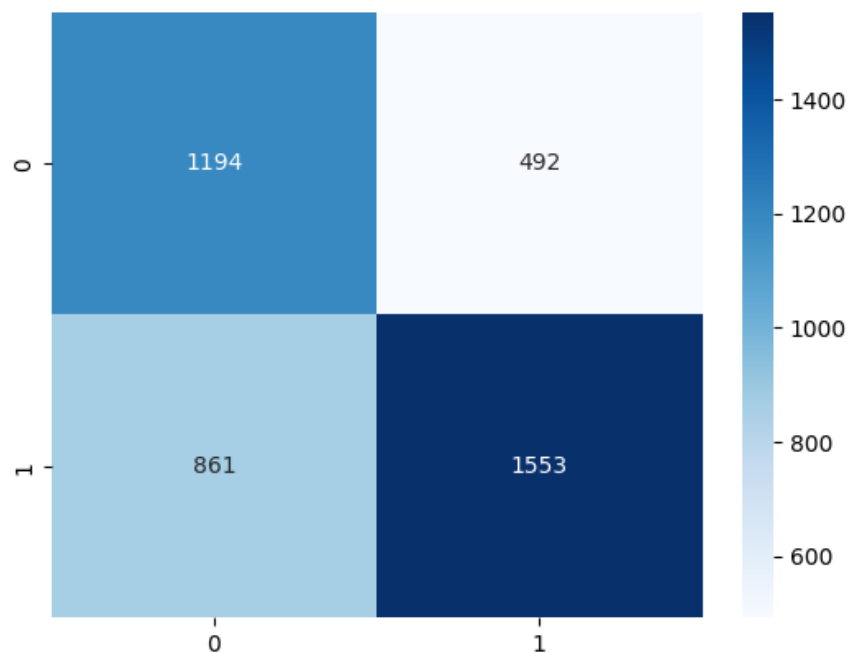
```

accuracy of logistic_regression is 67.00

```

	precision	recall	f1-score	support
0	0.71	0.58	0.64	2055
1	0.64	0.76	0.70	2045
accuracy			0.67	4100
macro avg	0.68	0.67	0.67	4100
weighted avg	0.68	0.67	0.67	4100

شکل ۴۴ - پارامترهای ارزیابی در مدل *LogisticRegression* پس از اعمال *PCA* زمانی که ابعاد پیکسل‌ها ۱۰۰ در ۱۰۰ است



شکل ۴۵ - ماتریس درهم ریختگی در مدل Logistic Regression ماننی که ابعاد پیکسل ها ۱۰۰ در ۱۰۰ است.

Logistic Regression with PCA

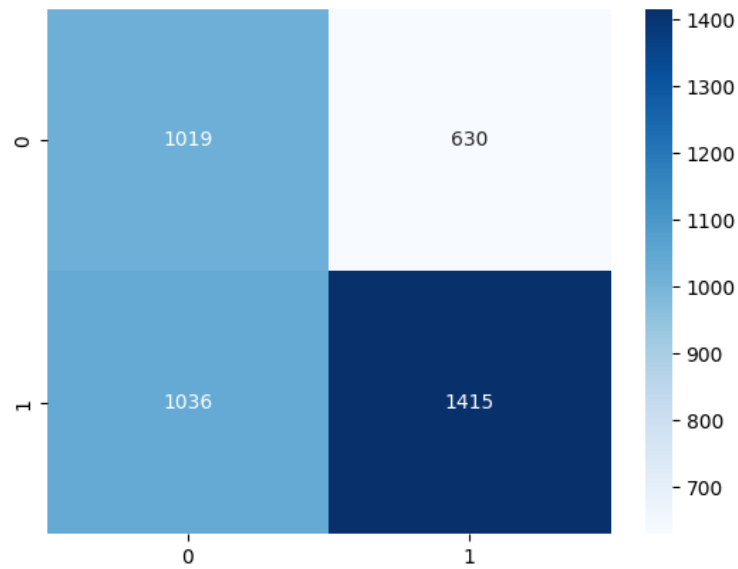
```

accuracy of Gaussian is 59.37
precision recall f1-score support
0 0.62 0.50 0.55 2055
1 0.58 0.69 0.63 2045

accuracy 0.59 4100
macro avg 0.60 0.59 0.59 4100
weighted avg 0.60 0.59 0.59 4100

```

شکل ۴۶ - پارامترهای ارزیابی در مدل LogisticRegression پس از اعمال PCA زمانی که ابعاد پیکسل ها ۱۰۰ در ۱۰۰ است

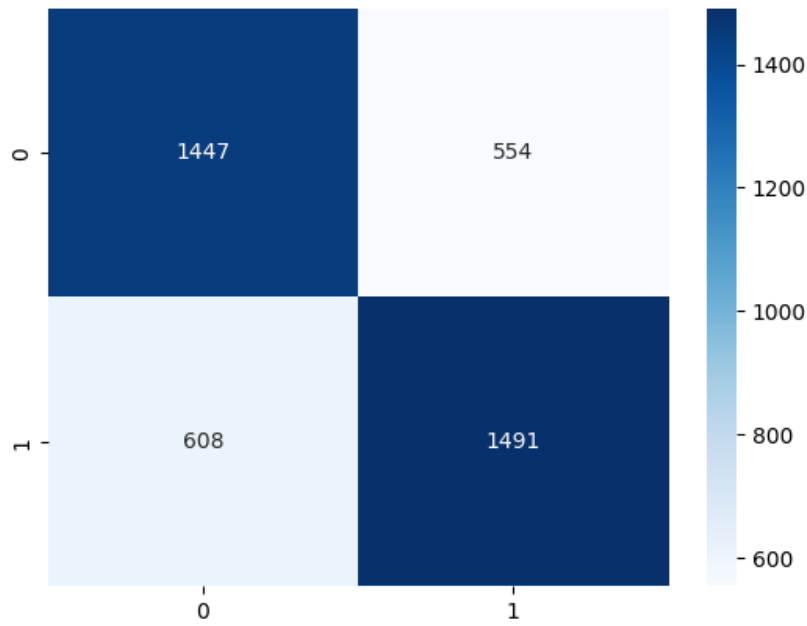


شکل 47- ماتریس درهم ریختگی در مدل Logistic Regression پس از اعمال PCA مانعی که ابعاد پیکسل ها ۱۰۰ در ۱۰۰ است.

MLP

accrracy of Gaussian is 71.66					
	precision	recall	f1-score	support	
0	0.72	0.70	0.71	2055	
1	0.71	0.73	0.72	2045	
accuracy			0.72	4100	
macro avg	0.72	0.72	0.72	4100	
weighted avg	0.72	0.72	0.72	4100	

شکل 48- پارامترهای ارزیابی در مدل MLP پس از اعمال PCA زمانی که ابعاد پیکسل ها ۱۰۰ در ۱۰۰ است



شکل ۴۹ - ماتریس درهم ریختگی در مدل MLP زمانی که ابعاد پیکسل ها ۱۰۰ در ۱۰۰ است پس از اعمال PCA

Decision Tree

```

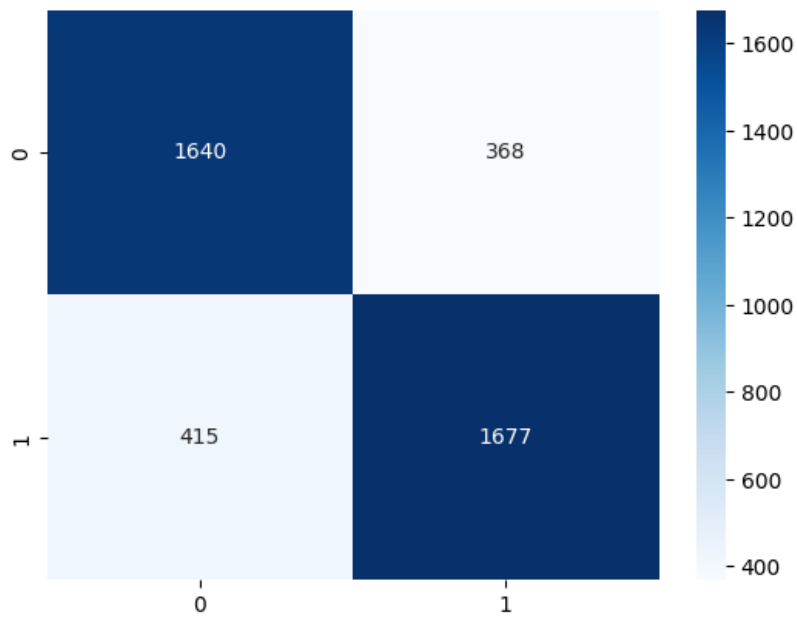
accuracy of Gaussian is 85.15
precision    recall  f1-score   support

      0       0.85      0.85      0.85     2055
      1       0.85      0.85      0.85     2045

 accuracy          0.85     4100
 macro avg       0.85      0.85      0.85     4100
 weighted avg    0.85      0.85      0.85     4100

```

شکل ۵۰ - پارامترهای ارزیابی در مدل decision tree زمانی که ابعاد پیکسل ها ۱۰۰ در ۱۰۰ است



شکل ۵۱ - ماتریس درهم ریختگی در مدل decision tree زمانی که ابعاد پیکسل ها ۱۰۰ در ۱۰۰ است

Decision Tree with PCA

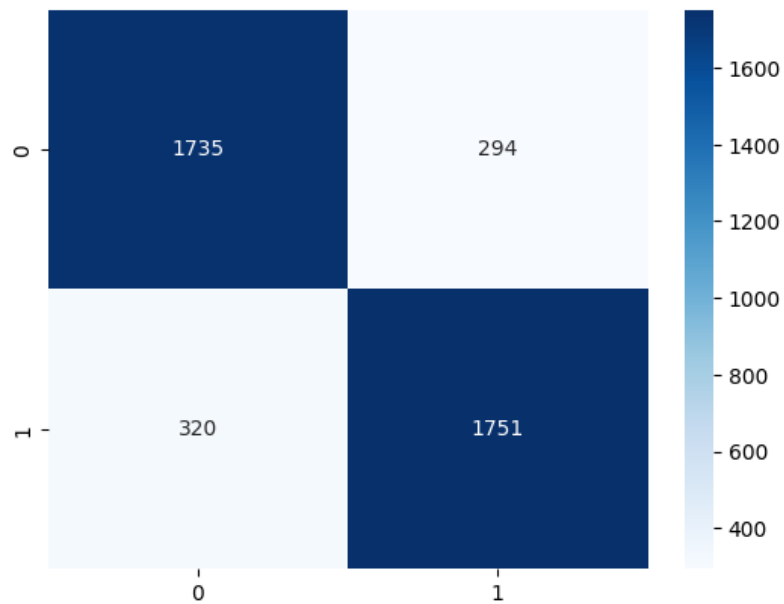
```

accrracy of Gaussian is 72.61
      precision    recall  f1-score   support

      0       0.73      0.71      0.72      2055
      1       0.72      0.74      0.73      2045

 accuracy          0.73          4100
 macro avg         0.73          0.73          0.73          4100
 weighted avg      0.73          0.73          0.73          4100
  
```

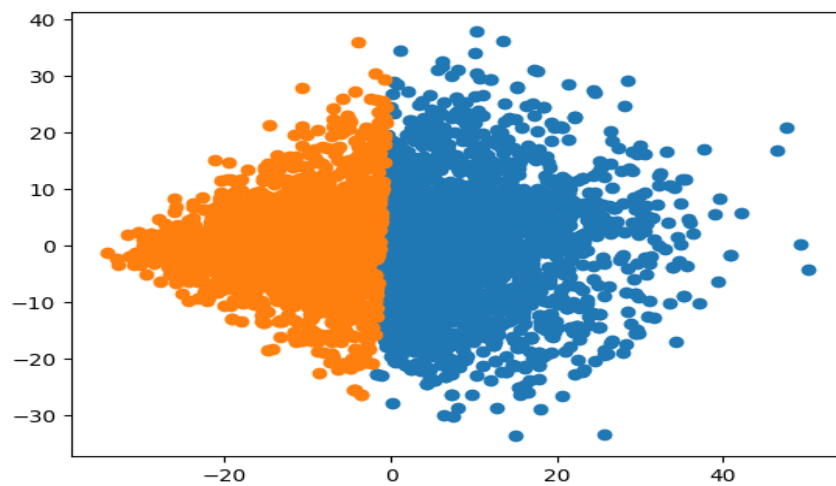
شکل ۵۲ - پارامترهای ارزیابی در مدل decision tree پس از اعمال PCA زمانی که ابعاد پیکسل ها ۱۰۰ در ۱۰۰ است



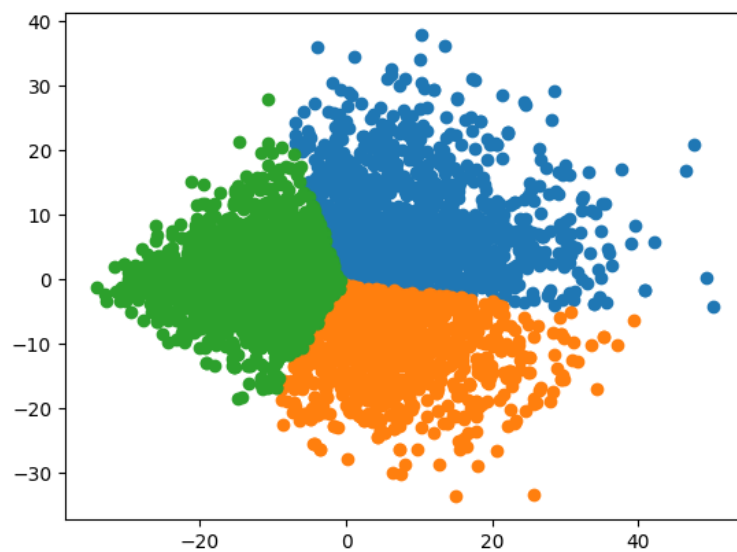
شکل ۵۳- ماتریس درهم ریختگی در مدل پس از اعمال PCA decision tree زمانی که ابعاد پیکسل ها ۱۰۰ در ۱۰۰ است

خوشه بندی داده ها بر حسب ویژگی های داده شده

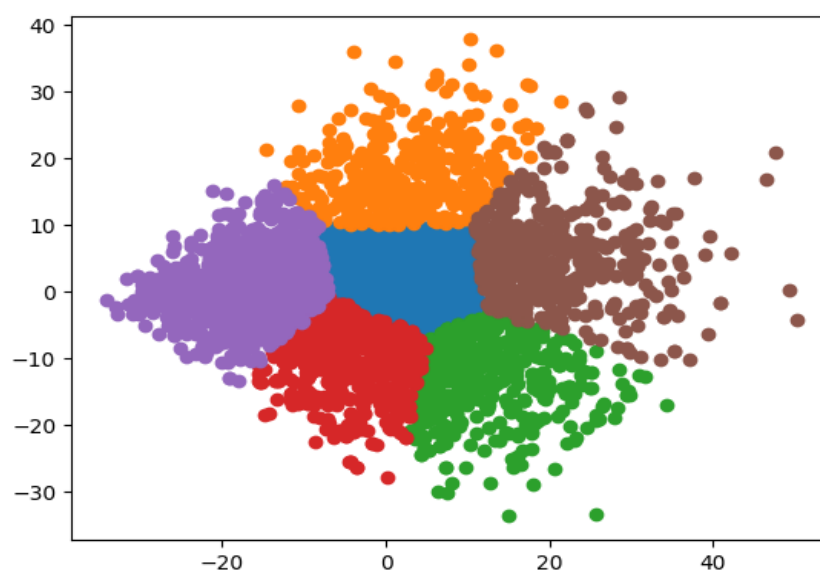
K-means



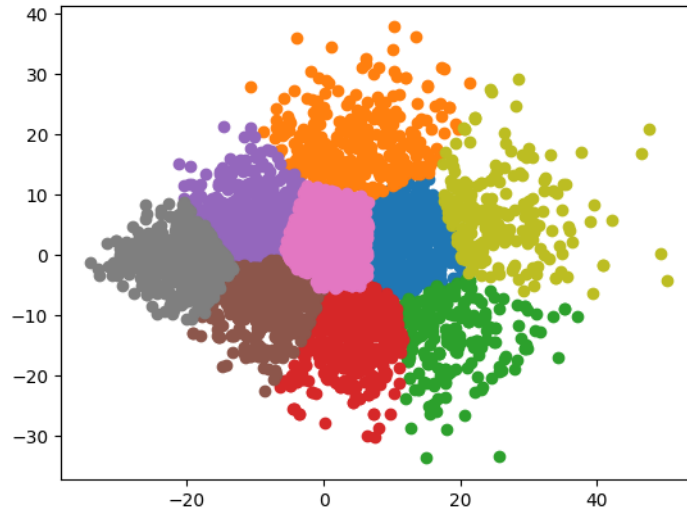
شکل ۵۴ - الگوریتم Kmeans با ۲ اجزا



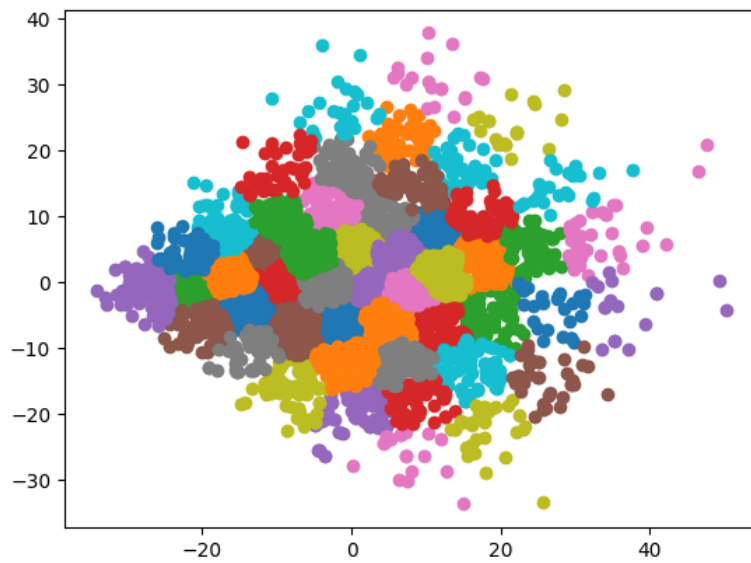
شکل ۵۵ - الگوریتم Kmeans با ۳ اجزا



شکل ۵۶ - الگوریتم Kmeans با ۶ اجزا

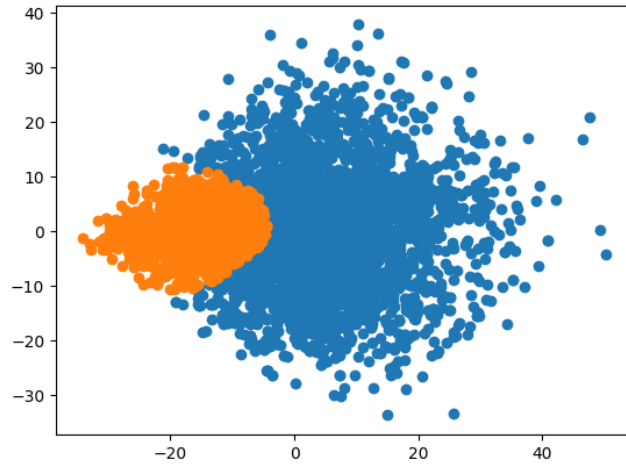


شكل ٥٧- الگوریتم Kmeans با ٩ اجزا

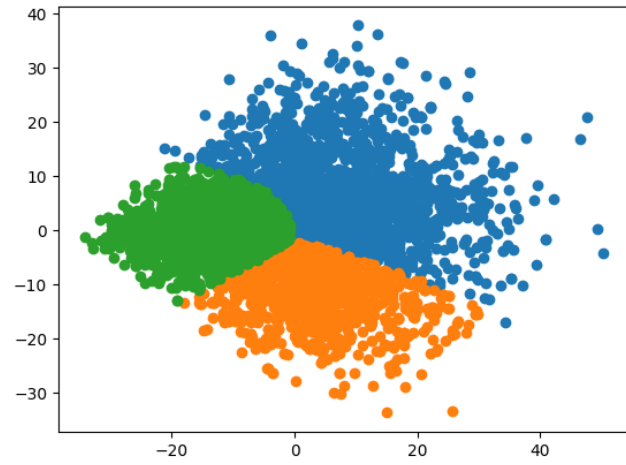


شكل ٥٨- الگوریتم Kmeans با ٥٠ اجزا

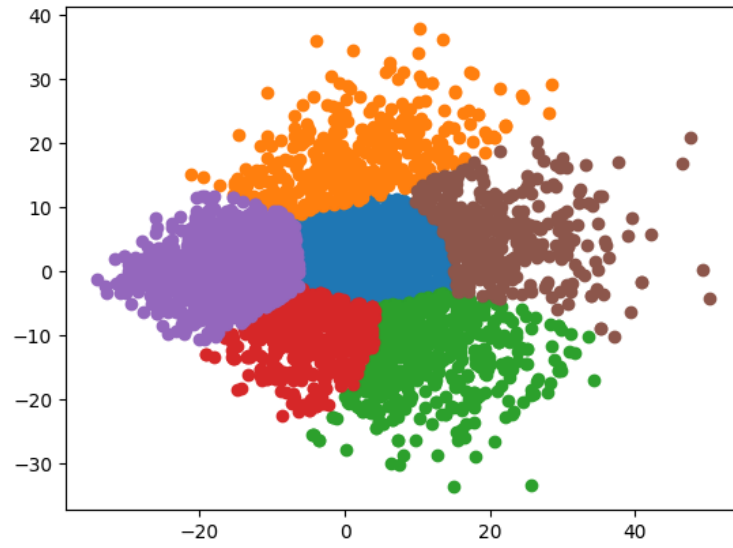
GMM



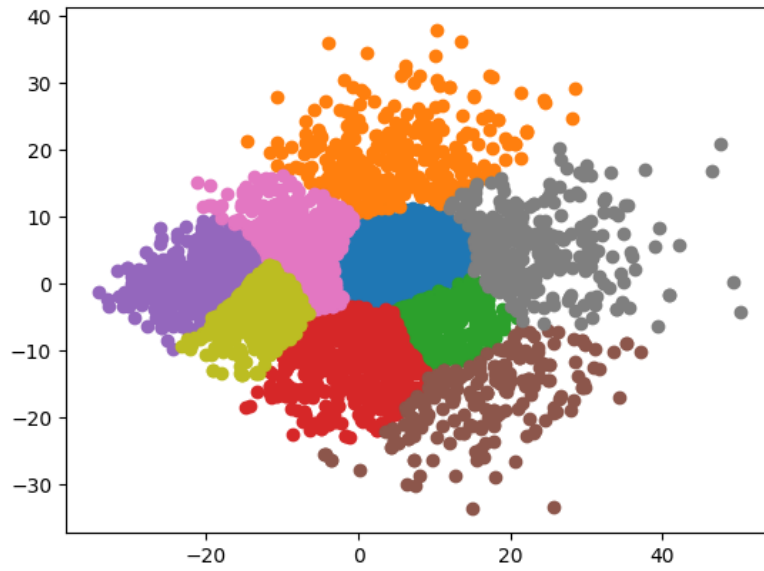
شکل ۵۹ - الگوریتم Kmeans با ۲ اجزا



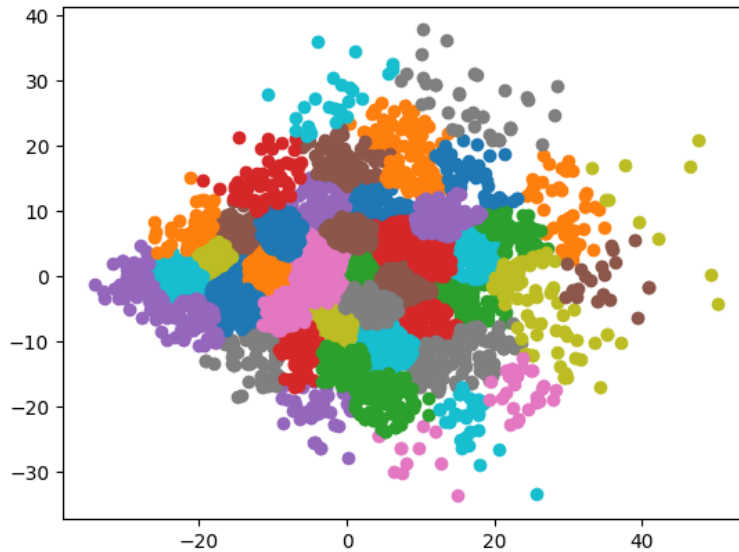
شکل ۶۰ - الگوریتم Kmeans با ۳ اجزا



شكل ٦١ - الگوریتم Kmeans با ٦ اجزا



شكل ٦٢ - الگوریتم Kmeans با ٩ اجزا



شکل ۶۳ - الگوریتم Kmeans با ۵۰ اجزا

در این قسمت و قسمتی که در قبل با استفاده از ویژگی های آماده تولید کردیم می توانیم با متریک های متفاوتی این دو را به جهت خوشه بندی مقایسه کرد در این جا با استفاده از متریک فاصله Intra-Cluster و Inter-Cluster می کنیم.

طبیعتاً در زمانی که از فیچر های آماده استفاده می کردیم، فاصله بین کلاس ها بیشتر از زمانی است که از فیچر های استخراجی خودمان استفاده کردیم. که با تصویر سازی داده ها این بسیار مشهود است همین طور فاصله بین خود داده ها در زمانی که از فیچر های آماده استفاده کردیم بسیار کمتر است که کم بودن فاصله در هر کلاس و زیاد بودن فاصله کلاس ها از هم مطلوب است.