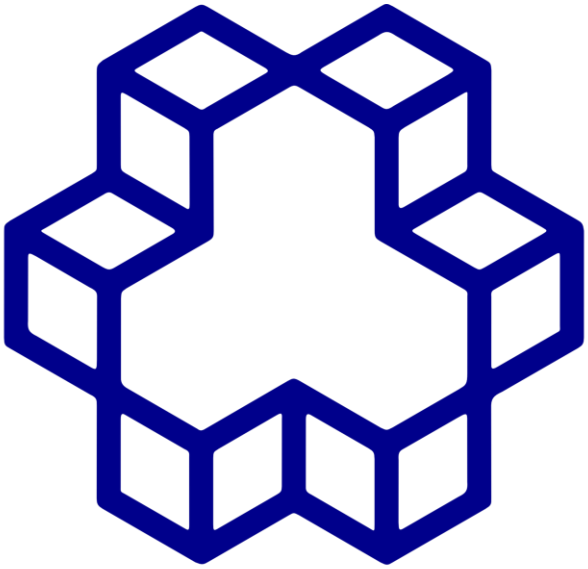


دانشگاه صنعتی خواجه نصیرالدین طوسی

ML COURSE

MINI PROJECT 1

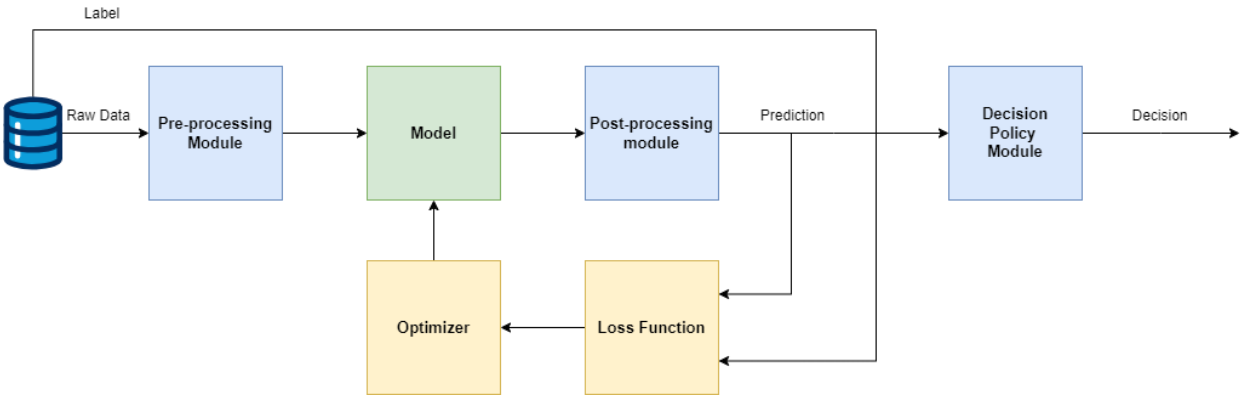
LINKS

Google Drive

GitHub

Mohammad Hoseyni
9821253

سؤال ۱



۱. فرآیند آموزش و ارزیابی یک مدل طبقه بند خطی را به صورت دیاگرامی بلوکی نمایش دهید و در مورد اجزای مختلف این دیاگرام بلوکی توضیحاتی بنویسید. تغییر نوع طبقه بندی از حالت دوکلاسه به چندکلاسه در کدام قسمت از این دیاگرام بلوکی تغییراتی ایجاد می کند؟ توضیح دهید.

Dataset:این قسمت شامل داده‌های خام و دسته‌بندی متناظر با آن است. کیفیت و کمیت مجموعه‌داده به‌صورت مؤثری روی عملکرد مدل یادگیری ماشین تأثیر خواهد گذاشت.

Pre-Processing Module: این بخش از ساختار یادگیری ماشین داده‌های خام را دریافت می‌کند و متناسب با وظیفه (Task) ساختار و نوع داده ورودی، با انجام انواع گوناگون تبدیل‌ها (Transforms) مانند حذف داده‌های پرت، استخراج ویژگی و فیلترگذاری مناسب داده‌ها را برای پردازش و آموزش الگوریتم یادگیری ماشین بهبود می‌بخشد. بخش پیش‌پردازش یک ساختار یادگیری ماشین می‌تواند روی کیفیت و کمیت یک مجموعه‌داده تأثیرگذار باشد، به‌عنوان نمونه با انجام تبدیل‌های ساخت داده مصنوعی (Augmentation) می‌توان تا حدی کمیت مجموعه‌داده را جبران کرد و با استفاده از این تبدیل‌ها به‌صورت وزن‌دار می‌توان برخی از مشکلات کیفی مجموعه‌داده مانند عدم تعادل (imbalance) را برطرف کرد.

Model: این بخش از ساختار یادگیری ماشین شامل الگوریتم یادگیری ماشین است که باتوجه‌به وظیفه (Regression, Classification, Segmentation) و ویژگی‌های متفاوت مجموعه‌داده مانند کیفیت و بار محاسباتی مطلوب، می‌تواند از میان انواع گوناگون انتخاب شود.

Post-Processing Module: این بخش از ساختار یادگیری ماشین شامل روش‌هایی است که باتوجه‌به خروجی الگوریتم یادگیری ماشین می‌تواند انتخاب شود تا پیش‌بینی‌های اشتباهی که به‌صورت آماری از یک الگوی خاص پیروی می‌کنند اصلاح شوند.

Loss Function: این بخش از ساختار یادگیری ماشین شامل انواع گوناگونی از توابع هزینه مانند MSE، Cross-Entropy و یا انواع دیگر توابع هزینه می‌شود. تابع هزینه خصوصاً برای الگوریتم‌هایی که شامل بهینه‌سازی از طریق مشتق هستند دارای اهمیت ویژه‌ای است؛ زیرا توابع هزینه‌ای که مشتق‌پذیر نیستند یا مشتق آنها در نواحی خاصی خیلی زیاد یا خیلی کم است برای فرایند آموزش مشکل‌ساز خواهند بود. تابع هزینه می‌تواند بعضی از مشکلات کیفی مجموعه‌داده مانند عدم تعادل را برطرف کند.

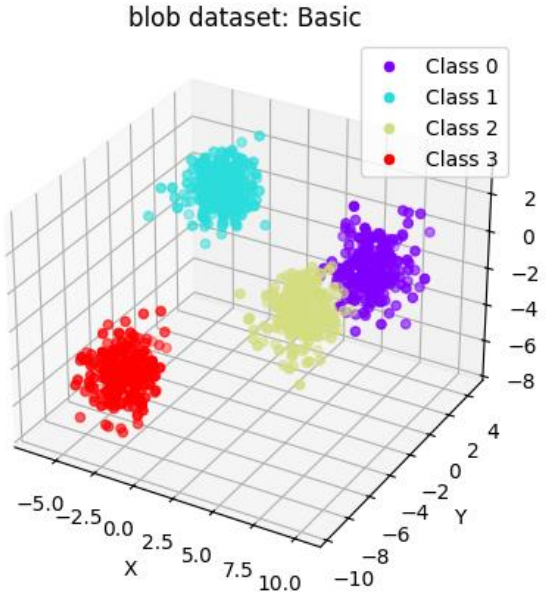
Optimizer: این قسمت از ساختار یادگیری ماشین شامل راه‌حل ریاضی برای بهینه‌سازی وزن‌های داخلی مدل یادگیری ماشین است. با درنظرگرفتن ویژگی‌های مجموعه‌داده و انتخاب صحیح الگوریتم بهینه‌سازی می‌توانیم زمان آموزش و عملکرد ساختار یادگیری ماشین را بهبود ببخشیم. یک الگوریتم بهینه‌سازی مناسب باعث می‌شود تا مدل یادگیری ماشین از نقاط بهینه محلی عبور کند و به نقاطی برود که تابع هزینه برای آنها مقدار کمتری را نشان می‌دهد.

Decision Policy Module: این قسمت از ساختار یادگیری ماشین باتوجه‌به وظیفه ساختار، یکی از انواع تبدیل‌های موجود را روی خروجی پیاده می‌کند. این تبدیل برای وظیفه طبقه‌بندی (Classification) می‌تواند تبدیل خروجی احتمالاتی به خروجی دو وضعیتی باشد که از آن با عنوان One-Hot Transform یاد می‌شود. همچنین در این بخش معیارهای ارزیابی ساختار مانند Recall، Precision و یا Accuracy محاسبه می‌شود.

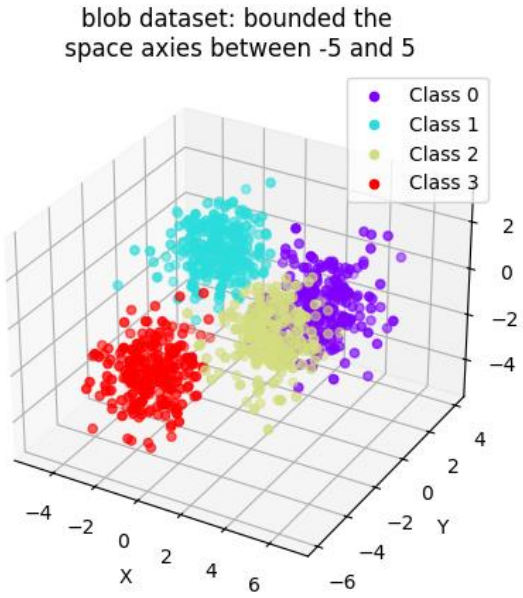
به‌منظور تغییر طبقه‌بندی از ۲ کلاس به چند کلاس، می‌تواند در تمام قسمت‌های ساختار تأثیر بگذارد بااین‌حال مهم‌ترین قسمت شامل Loss Function و Optimizer است که باید برای استفاده به‌عنوان یک طبقه‌بند چند کلاسه با شرایط خروجی مدل سازگاری داشته باشند. به‌عنوان نمونه BCELoss در کتابخانه PyTorch به‌منظور استفاده در یک مسئله طبقه‌بندی چند کلاسه مطلوب نیست و همچنین در مدل LogisticRegression از کتابخانه sklearn اگر برای وظیفه طبقه‌بندی چند کلاسه استفاده شود، نمی‌تواند از liblinear به‌عنوان solver استفاده کند. در انتها به‌منظور تبدیل مدل از طبقه‌بند دو کلاسه به چند کلاسه می‌توان دو رویکرد multinomial یا one-vs-rest استفاده کرد که هنگام استفاده از رویکرد one-vs-rest، وظیفه یک طبقه‌بندی چند کلاسه به چندوظیفه طبقه‌بندی دو کلاسه تبدیل می‌شود و محدودیت‌های فوق‌الذکر برای این رویکرد وجود ندارد.

۲. با استفاده از datasets.sklearn ، یک دیتاست با ۱۰۰۰ نمونه، ۴ کلاس و ۳ ویژگی تولید کنید و آن را به صورتی مناسب نمایش دهید. آیا دیتاستی که تولید کردید چالش برانگیز است؟ چرا؟ به چه طریقی می توانید دیتاست تولیدشده خود را چالش برانگیزتر و سخت تر کنید؟

به‌منظور ساخت یک مجموعه‌داده طبقه‌بندی از کتابخانه sklearn و متد make_blobs استفاده شده است. نتیجه ساخت یک مجموعه‌داده متعادل با ۴ کلاس و ۳ ویژگی با تنظیمات پیش‌فرض متد make_blobs در تصویر زیر نمایش داده شده است.



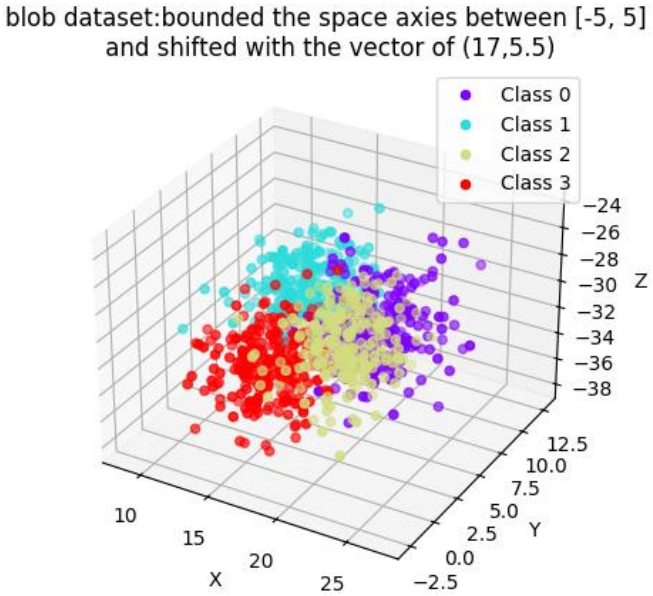
همان‌طور که از این تصویر مشخص است تفکیک‌پذیری داده‌ها در این مجموعه‌داده پالش برانگیز نمی‌باشد و تنها چالش این مجموعه‌داده جداسازی نقاط بنفش (class0) از نقاط زرد (class2) است. در ادامه به‌منظور چالش‌برانگیز کردن این مجموعه‌داده از میان پارامترهای قابل‌تنظیم در متد make_blobs، center_box= [-5, 5] قرار داده می‌شود. نتیجه این تنظیمات



مجموعه‌داده زیر است.

در این مجموعه‌داده با کم‌شدن محدوده مجاز حضور نقاط در فضا، باعث نزدیک‌شدن مرکز دسته‌های هر کلاسه به یکدیگر شده است و در نتیجه آن چالش جداسازی نقاط از هم و تشخیص درجه تعلق آنها به هر دسته چالش‌برانگیزتر شده است؛ اما همچنان جداسازی این داده‌ها از یکدیگر می‌تواند چالش‌برانگیزتر باشد. با تغییر انحراف از معیار مربوط به دستور ساخت مجموعه‌داده می‌توان چالش این مجموعه‌داده را افزایش داد. در ادامه انحراف از معیار مجموعه‌داده برابر ۲ قرار داده شده است و نتیجه آن در شکل زیر قابل مشاهده است.

در مجموعه‌داده فوق تفکیک داده‌ها از یکدیگر بسیار مشکل شده است و می‌تواند وظیفه‌ای چالش‌برانگیز برای هر ساختار یادگیری ماشین باشد. به‌منظور اضافه‌کردن نامعینی می‌توان نویز را به نقاط مجموعه‌داده اضافه کرد که در ادامه این آزمایش از انجام آن



صرف‌نظر شده است. در ادامه مجموعه‌داده فوق با مقادیر ثابت [17, 5.5, -30.33] جمع شده است تا تاثیر normalization مجموعه داده قابل تشخیص باشد.

۳. با استفاده از حداقل دوطبقه بند خطی آمادهٔ پایتون (در model_linear.sklearn) و در نظر گرفتن فراپارامترهای مناسب، چهار کلاس موجود در دیتاست قسمت قبلی را از هم تفکیک کنید. ضمن توضیح روند انتخاب فراپارامترها (مانند تعداد دورهٔ آموزش و نرخ یادگیری)، نتیجهٔ دقت آموزش و ارزیابی را نمایش دهید. برای بهبود نتیجه از چه تکنیک هایی استفاده کردید؟

در ادامه از دو طبقه‌بند خطی LogisticRegression و SGDClassifier استفاده شده است و با استفاده از کتابخانه sklearn پیاده‌سازی آن انجام شده است. این نکته حائز اهمیت است که ۲۰ درصد از مجموعه داده مذکور برای انجام عمل ارزیابی و ۸۰ درصد برای عمل آموزش جدا شده است.

در ابتدا این مدل روی مجموعه‌داده خام آموزش و به‌ازای دو رویکرد متفاوت multinomial و ovr و تمام solverهای موجود آموزش‌دیده

```
These results are for training the LogisticRegression model before apply the normalization transform on the training and test set.
c:\Users\mamdaliof\anaconda3\envs\ML\lib\site-packages\sklearn\linear_model\_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge

c:\Users\mamdaliof\anaconda3\envs\ML\lib\site-packages\sklearn\linear_model\_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge

model with sag solver and multi_class=multinomial score on train set is equal ot 0.7775
model with sag solver and multi_class=multinomial score on test set is equal ot 0.735
test set: accuracy = 0.735, recall = 0.735, precision = 0.735

model with saga solver and multi_class=multinomial score on train set is equal ot 0.77125
model with saga solver and multi_class=multinomial score on test set is equal ot 0.72
test set: accuracy = 0.72, recall = 0.72, precision = 0.72

model with lbfgs solver and multi_class=multinomial score on train set is equal ot 0.79375
model with lbfgs solver and multi_class=multinomial score on test set is equal ot 0.735
test set: accuracy = 0.735, recall = 0.735, precision = 0.735

model with newton-cg solver and multi_class=multinomial score on train set is equal ot 0.79375
model with newton-cg solver and multi_class=multinomial score on test set is equal ot 0.735
test set: accuracy = 0.735, recall = 0.735, precision = 0.735


 ------------------------------------------------------------
```

است که نتایج آن به شرح زیر است.

```
c:\Users\mamdaliof\anaconda3\envs\ML\lib\site-packages\sklearn\linear_model\_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge

c:\Users\mamdaliof\anaconda3\envs\ML\lib\site-packages\sklearn\linear_model\_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge

c:\Users\mamdaliof\anaconda3\envs\ML\lib\site-packages\sklearn\linear_model\_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge

c:\Users\mamdaliof\anaconda3\envs\ML\lib\site-packages\sklearn\linear_model\_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge

model with sag solver and multi_class=ovr score on train set is equal ot 0.78625
model with sag solver and multi_class=ovr score on test set is equal ot 0.745
test set: accuracy = 0.745, recall = 0.745, precision = 0.745

model with saga solver and multi_class=ovr score on train set is equal ot 0.78125
model with saga solver and multi_class=ovr score on test set is equal ot 0.73
test set: accuracy = 0.73, recall = 0.73, precision = 0.73

model with lbfgs solver and multi_class=ovr score on train set is equal ot 0.7825
model with lbfgs solver and multi_class=ovr score on test set is equal ot 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75

model with liblinear solver and multi_class=ovr score on train set is equal ot 0.76125
model with liblinear solver and multi_class=ovr score on test set is equal ot 0.725
test set: accuracy = 0.725, recall = 0.725, precision = 0.725

model with newton-cg solver and multi_class=ovr score on train set is equal ot 0.7825
model with newton-cg solver and multi_class=ovr score on test set is equal ot 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75

model with newton-cholesky solver and multi_class=ovr score on train set is equal ot 0.7825
model with newton-cholesky solver and multi_class=ovr score on test set is equal ot 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75
```

همان‌طور که مشخص است تعدادی از الگوریتم‌ها همگرا نشده‌اند و نتیجه این آموزش از لحاظ بار محاسباتی و دقت مناسب نمی‌باشد.

در گام بعد تبدیل Normalization روی مجموعه‌داده انجام می‌شود و نتایج آموزش به شرح زیر است.

```
These results are obtained from training LogisticRegression model with different solvers and different multi_class arguments

model with sag solver and multi_class=multinomial score on train set is equal to 0.79375
model with sag solver and multi_class=multinomial score on test set is equal to 0.74
test set: accuracy = 0.74, recall = 0.74, precision = 0.74

model with saga solver and multi_class=multinomial score on train set is equal to 0.79375
model with saga solver and multi_class=multinomial score on test set is equal to 0.74
test set: accuracy = 0.74, recall = 0.74, precision = 0.74

model with lbfgs solver and multi_class=multinomial score on train set is equal to 0.79375
model with lbfgs solver and multi_class=multinomial score on test set is equal to 0.74
test set: accuracy = 0.74, recall = 0.74, precision = 0.74

model with newton-cg solver and multi_class=multinomial score on train set is equal to 0.79375
model with newton-cg solver and multi_class=multinomial score on test set is equal to 0.74
test set: accuracy = 0.74, recall = 0.74, precision = 0.74

------------------------------------------------------------
```

```
model with sag solver and multi_class=ovr score on train set is equal to 0.78125
model with sag solver and multi_class=ovr score on test set is equal to 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75

model with saga solver and multi_class=ovr score on train set is equal to 0.78125
model with saga solver and multi_class=ovr score on test set is equal to 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75

model with lbfgs solver and multi_class=ovr score on train set is equal to 0.78125
model with lbfgs solver and multi_class=ovr score on test set is equal to 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75

model with liblinear solver and multi_class=ovr score on train set is equal to 0.78
model with liblinear solver and multi_class=ovr score on test set is equal to 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75

model with newton-cg solver and multi_class=ovr score on train set is equal to 0.78125
model with newton-cg solver and multi_class=ovr score on test set is equal to 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75

model with newton-cholesky solver and multi_class=ovr score on train set is equal to 0.78125
model with newton-cholesky solver and multi_class=ovr score on test set is equal to 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75
```

در ادامه بین بازه ۰.۰۱ تا ۱.۵ با گام ۰.۰۱ مقادیر مختلفی برای پارامتر C که میزان Regularization را در فرایند آموزش تعیین می‌کند بررسی شده است تا بهترین مقدار محاسبه شود. نتایج این مجموعه آموزش به شرح زیر است:

```
These results indicate the beset value of C between 0.01 and 1.5

model with sag solver and multi_class=multinomial score on train set is equal to 0.78625. C=0.02
model with sag solver and multi_class=multinomial score on test set is equal to 0.74
test set: accuracy = 0.74, recall = 0.74, precision = 0.74

model with saga solver and multi_class=multinomial score on train set is equal to 0.78625. C=0.02
model with saga solver and multi_class=multinomial score on test set is equal to 0.74
test set: accuracy = 0.74, recall = 0.74, precision = 0.74

model with lbfgs solver and multi_class=multinomial score on train set is equal to 0.78625. C=0.02
model with lbfgs solver and multi_class=multinomial score on test set is equal to 0.74
test set: accuracy = 0.74, recall = 0.74, precision = 0.74

model with newton-cg solver and multi_class=multinomial score on train set is equal to 0.78625. C=0.02
model with newton-cg solver and multi_class=multinomial score on test set is equal to 0.74
test set: accuracy = 0.74, recall = 0.74, precision = 0.74
```

```
model with sag solver and multi_class=ovr score on train set is equal to 0.78. C=0.19
model with sag solver and multi_class=ovr score on test set is equal to 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75

model with saga solver and multi_class=ovr score on train set is equal to 0.78. C=0.19
model with saga solver and multi_class=ovr score on test set is equal to 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75

model with lbfgs solver and multi_class=ovr score on train set is equal to 0.78. C=0.19
model with lbfgs solver and multi_class=ovr score on test set is equal to 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75

model with liblinear solver and multi_class=ovr score on train set is equal to 0.7825. C=0.060000000000000005
model with liblinear solver and multi_class=ovr score on test set is equal to 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75

model with newton-cg solver and multi_class=ovr score on train set is equal to 0.78. C=0.19
model with newton-cg solver and multi_class=ovr score on test set is equal to 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75

model with newton-cholesky solver and multi_class=ovr score on train set is equal to 0.78. C=0.19
model with newton-cholesky solver and multi_class=ovr score on test set is equal to 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75
```

در ادامه دو مدل از میان تمام مدل‌های موجود انتخاب شده و باقی آموزش‌ها بر روی این مدل‌ها صورت خواهد گرفت. فراپارامتر بعدی که مقدار بهینه آن باید محاسبه شود tol است. مقدار این فراپارامتر در بازه 10^{-6} تا 10^{-1}عوض شده است تا حداقل مقدار خطای قابل قبول برای مجموعه‌داده آموزش محاسبه شود. این آموزش‌ها در شرایطی انجام شده است که مقدار C از قسمت قبل بدست آمده است.

```
among different solvers and multi_class approaches the newton-cg solver with multinomial approach and liblinear solver with ovr selected and the process will be continued

model with newton solver and multi_class=multinomial score on train set is equal to 0.78625. tol=0.02
model with newton solver and multi_class=multinomial score on test set is equal to 0.74
test set: accuracy = 0.74, recall = 0.74, precision = 0.74

model with liblinear solver and multi_class=ovr score on train set is equal to 0.7825. tol=1e-06
model with liblinear solver and multi_class=ovr score on test set is equal to 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75
```

در انتها فراپارامتر dual برای solver liblinear فعال شده است تا تغییراتی که این فراپارامتر در آموزش ایجاد می‌کند مشاهده شود. نتایج حاصل شده در ادامه قابل‌مشاهده است.

```
The dual parameter is only applicable for liblinear solver thus the solver is liblinear and dual is true.
model with liblinear solver and multi_class=ovr score on train set is equal to 0.7825.
model with liblinear solver and multi_class=ovr score on test set is equal to 0.75
test set: accuracy = 0.75, recall = 0.75, precision = 0.75
```

حال مدل SGDClassifier به درنظرگرفتن lossهای موجود آموزش‌دیده شده است که نتایج آن در ادامه قابل‌مشاهده است. در ابتدا مدل روی داده‌های نرمال آموزش‌دیده است.

```
the SGDClassifier with different losses is calculated here

model with hinge loss function score is equal ot 0.765 on the train set
model with hinge loss function score is equal ot 0.72 on the test set
test set: accuracy = 0.72, recall = 0.72, precision = 0.72

model with log_loss loss function score is equal ot 0.7025 on the train set
model with log_loss loss function score is equal ot 0.665 on the test set
test set: accuracy = 0.665, recall = 0.665, precision = 0.665

model with perceptron loss function score is equal ot 0.69125 on the train set
model with perceptron loss function score is equal ot 0.655 on the test set
test set: accuracy = 0.655, recall = 0.655, precision = 0.655

model with squared_error loss function score is equal ot 0.30375 on the train set
model with squared_error loss function score is equal ot 0.295 on the test set
test set: accuracy = 0.295, recall = 0.295, precision = 0.295
```

همان‌طور که مشخص است بعضی از مدل‌ها عملکرد مناسبی از خود نشان نداده‌اند که در نتیجه از ادامه فرایند آموزش حذف خواهند شد. در ادامه فراپارامتر learning_rate='adaptive' قرار داده شده است و نتایج آموزش آن قابل نسبت به نتایج گذشته بهتر است.

```
Models with better performance selected and remains and the learning_rate is adaptive

model with hinge loss function score is equal ot 0.7675 on the train set
model with hinge loss function score is equal ot 0.705 on the test set
test set: accuracy = 0.705, recall = 0.705, precision = 0.705

model with log_loss loss function score is equal ot 0.77 on the train set
model with log_loss loss function score is equal ot 0.71 on the test set
test set: accuracy = 0.71, recall = 0.71, precision = 0.71

model with perceptron loss function score is equal ot 0.71375 on the train set
model with perceptron loss function score is equal ot 0.665 on the test set
test set: accuracy = 0.665, recall = 0.665, precision = 0.665
```

در ادامه مقدار اولیه eta0 در بازه ۰.۰۰۰۱ تا ۵۰ انتخاب خواهد شد و نتایج بهترین آموزش در ادامه قابل رویت است.

```
The best eta0 is selected between 1e13 and 20
model with hinge loss function score is equal ot 0.7875 on the train set. eta0=28.947789473684214
model with hinge loss function score is equal ot 0.74 on the test set. eta0=28.947789473684214
test set: accuracy = 0.74, recall = 0.74, precision = 0.74

model with log_loss loss function score is equal ot 0.7825 on the train set. eta0=2.6325263157894736
model with log_loss loss function score is equal ot 0.745 on the test set. eta0=2.6325263157894736
test set: accuracy = 0.745, recall = 0.745, precision = 0.745

model with perceptron loss function score is equal ot 0.77125 on the train set. eta0=2.632526315789473
model with perceptron loss function score is equal ot 0.71 on the test set. eta0=2.6325263157894736
test set: accuracy = 0.71, recall = 0.71, precision = 0.71
```

نتایج به‌دست‌آمده نشان‌دهنده بهبود فرایند آموزش است. در ادامه دو مدل که بهترین عملکرد را دارند انتخاب شده و باقی آموزش‌ها بروی آنها

انجام خواهد گرفت. آموزش بعدی به‌منظور به‌دست‌آوردن بهترین مقدار tol است که حداقل میزان قابل‌قبول برای تابع هزینه را روی مجموعه‌داده آموزش مشخص می‌کند.

```
model with hinge loss function score is equal ot 0.7875 on the train set. tol=1e-06
model with hinge loss function score is equal ot 0.74 on the test set. tol=1e-06
test set: accuracy = 0.74, recall = 0.74, precision = 0.74

model with log_loss loss function score is equal ot 0.7825 on the train set. tol=1e-06
model with log_loss loss function score is equal ot 0.745 on the test set. tol=1e-06
test set: accuracy = 0.745, recall = 0.745, precision = 0.745
```

برای به‌دست‌آوردن بهترین مقدار alpha که میزان regularization را مشخص می‌کند مقادیر آلفا را از 10^{-6} تا 10^{-2} جابجا می‌کنیم که نتیجه بهترین مقدار برای alpha در شکل زیر مشخص است.

```
The best value for alpha obtained between 1e-6 and 1e-2

model with hinge loss function score is equal ot 0.78625 on the train set. alpha=0.0010535263157894737
model with hinge loss function score is equal ot 0.74 on the test set. aplha=0.0010535263157894737
test set: accuracy = 0.74, recall = 0.74, precision = 0.74

model with log_loss loss function score is equal ot 0.7825 on the train set. alpha=1e-06
model with log_loss loss function score is equal ot 0.745 on the test set. aplha=1e-06
test set: accuracy = 0.745, recall = 0.745, precision = 0.745
```

در انتها بهترین نسبت مجموعه‌داده *valdidation* به‌ازای مقادیر ۰.۱و ۰.۱۵و ۰.۲ محاسبه شده است.

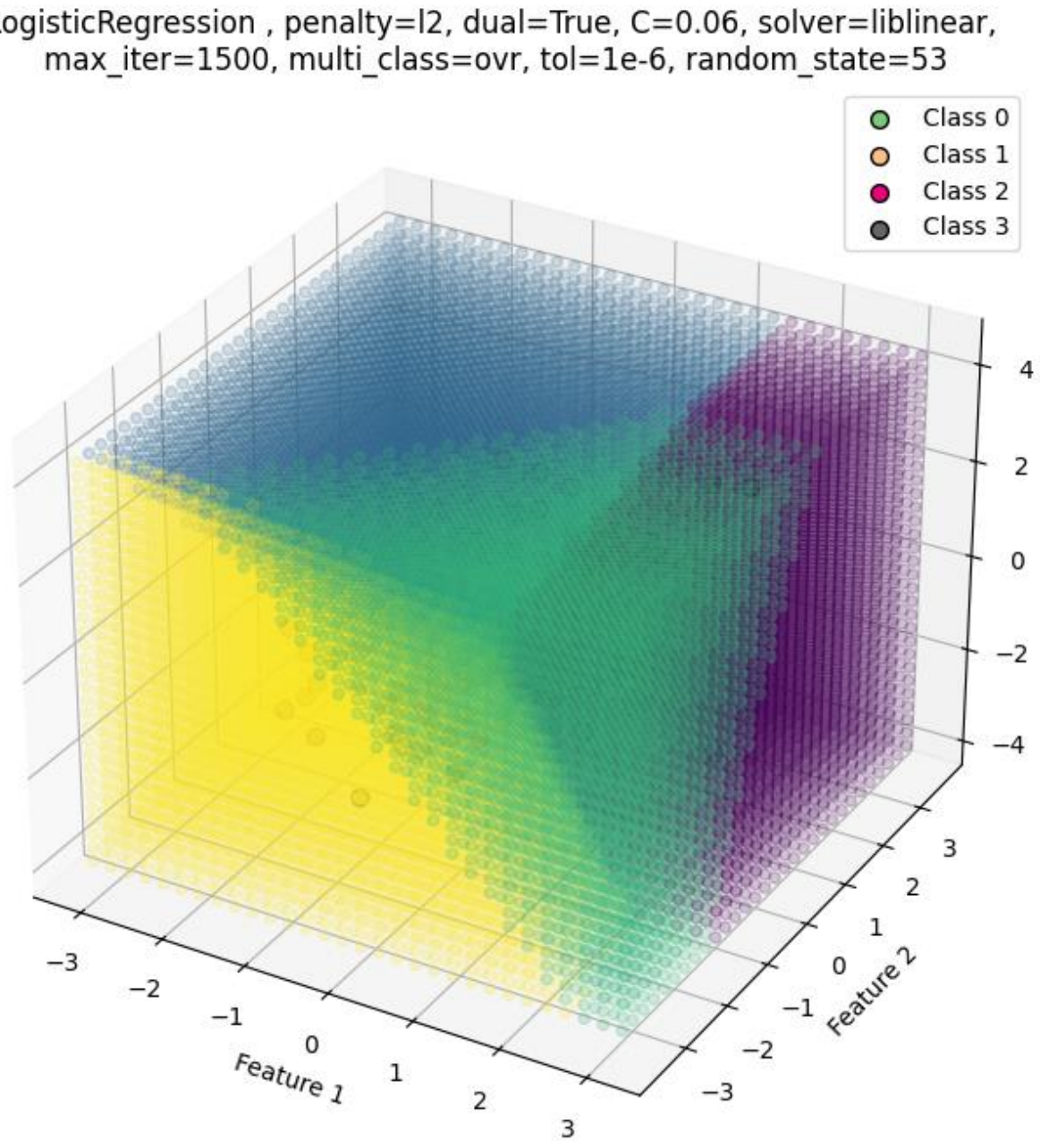
```
The best fraction ov train val is selected
model with hinge loss function score is equal ot 0.7875 on the train set. eta0=0.00105. validation_fraction=0.1
model with hinge loss function score is equal ot 0.74 on the test set. eta0=0.00105
test set: accuracy = 0.74, recall = 0.74, precision = 0.74

model with log_loss loss function score is equal ot 0.785 on the train set. eta0=1e-06. validation_fraction=0.1
model with log_loss loss function score is equal ot 0.745 on the test set. eta0=1e-06
test set: accuracy = 0.745, recall = 0.745, precision = 0.745
```

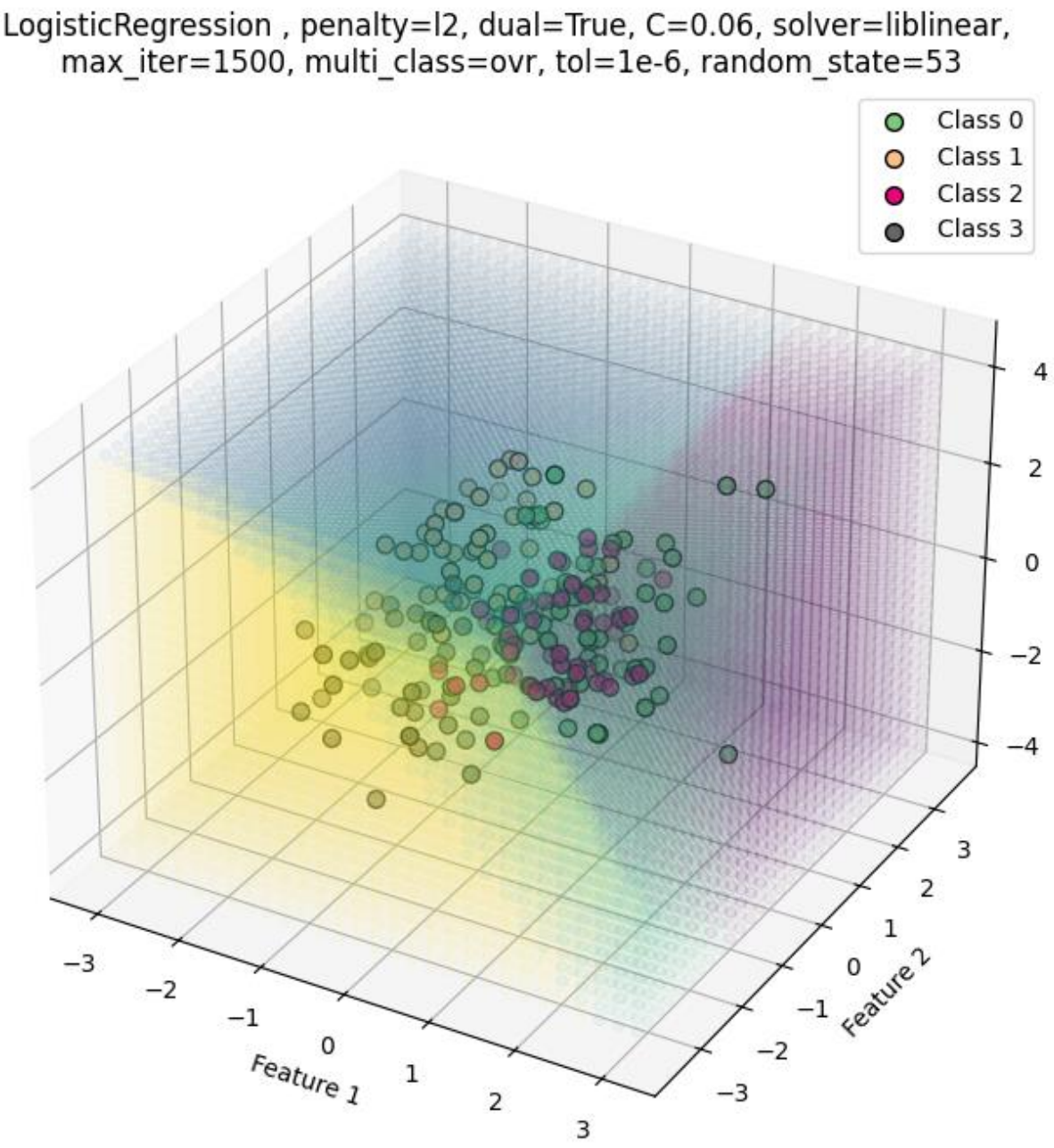
نکته مهم درمورد نتایج خروجی این است که در تصاویر فوق، مقادیر *precision* و*recall* به‌صورت *micro* محاسبه شده است که با مقدار *score* برابر است؛ اما در فایل *notebook* این مقادیر به‌صورت *macro* محاسبه شده است. در ادامه بهترین مدل بر اساس اطلاعات و *matric*های موجود روی مجموعه‌داده ارزیابی، مدلی است که به فراپارامترهای زیر آموزش ببینند.

```
LogisticRegression(penalty="l2",
                           dual=True,
                           C=0.06,
                           solver="liblinear",
                           max_iter=1500,
                           multi_class="ovr",
                           tol=1e-6,
                           random_state=53)
```

٤. مرز و نواحی تصمیم گیری برآمده از مدل آموزش دیدهٔ خود را به همراه نمونه ها در یک نمودار نشان دهید. اگر می توانید نمونه هایی که اشتباه طبقه بندی شده اند را با شکل و رنگ متفاوت نمایش دهید.

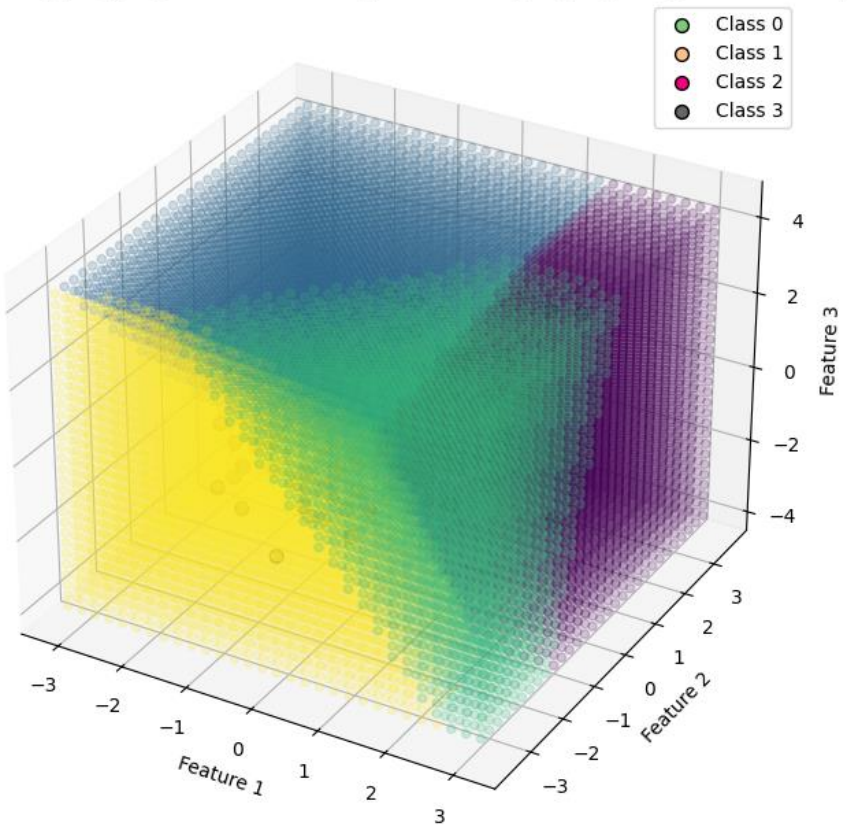
بدین منظور از ChatGPT کمک گرفته شده است. مرز تصمیم و نتایج پیش‌بینی‌های بهترین مدل روی مجموعه داده ارزیابی به شرح زیر است.



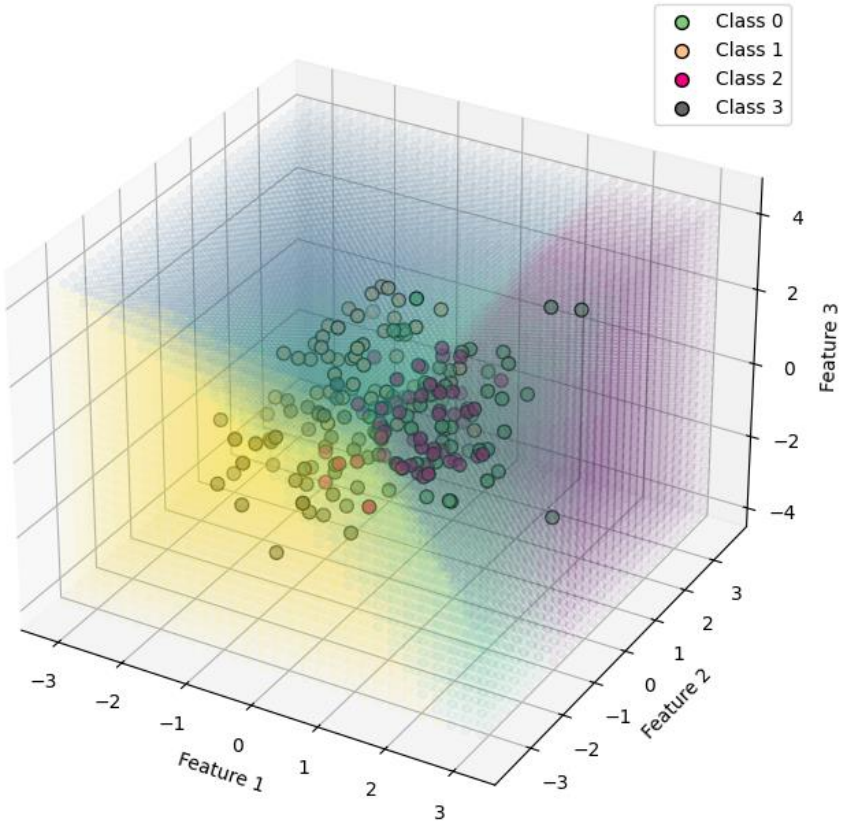


نمایش این نمودار به‌صورت interactive در داخل فایل Q1.1.ipynb قابل رویت است.

SGDClassifier, penalty=l2, loss='log_loss', alpha=1e-6, max_iter=4000, tol=1e-6, learning_rate='adaptive'
, eta0=2.632526, early_stopping=True, validation_fraction=i, n_iter_no_change=10, random_state=53

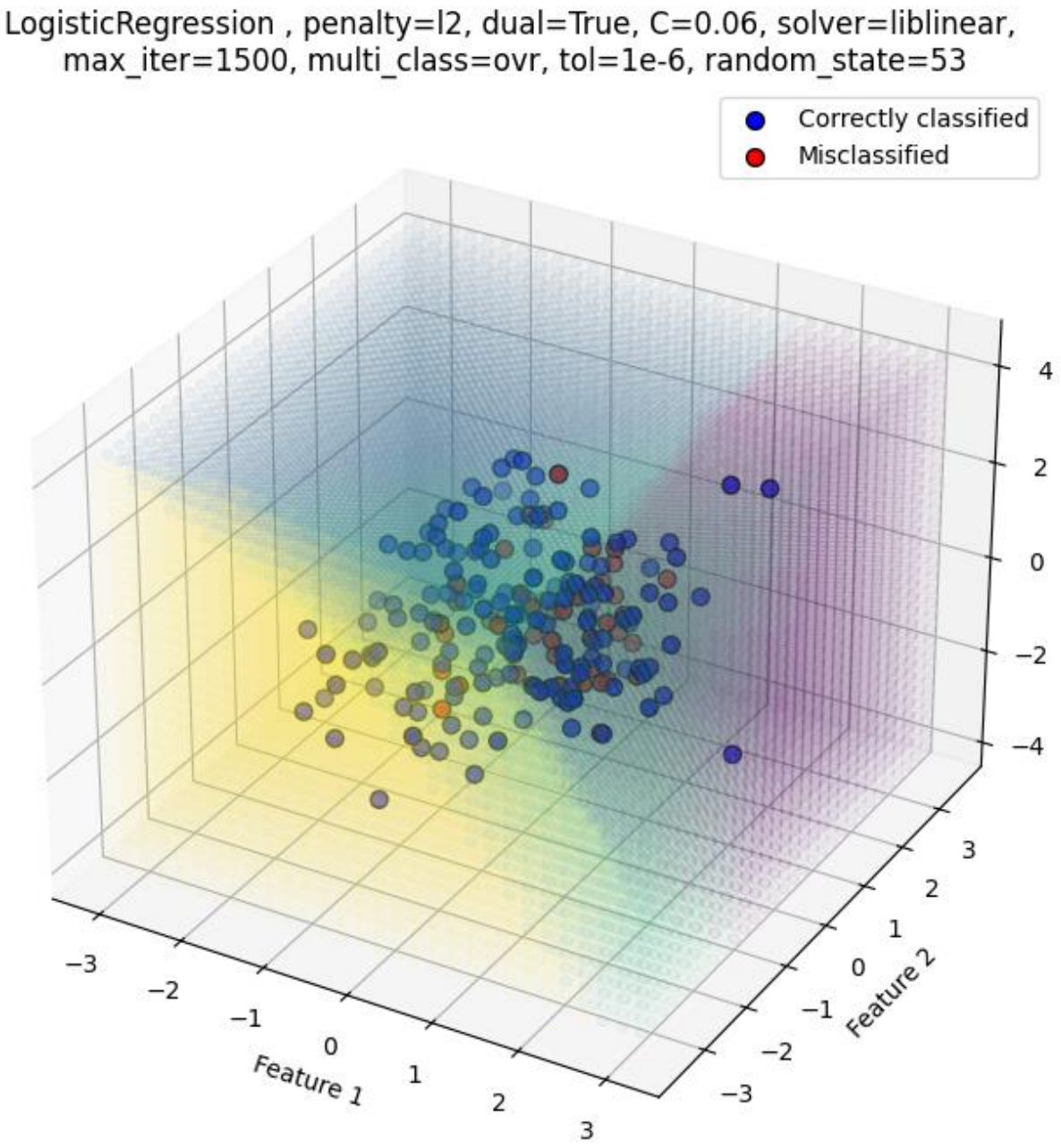


SGDClassifier, penalty=l2, loss='log_loss', alpha=1e-6, max_iter=4000, tol=1e-6, learning_rate='adaptive'
, eta0=2.632526, early_stopping=True, validation_fraction=i, n_iter_no_change=10, random_state=53

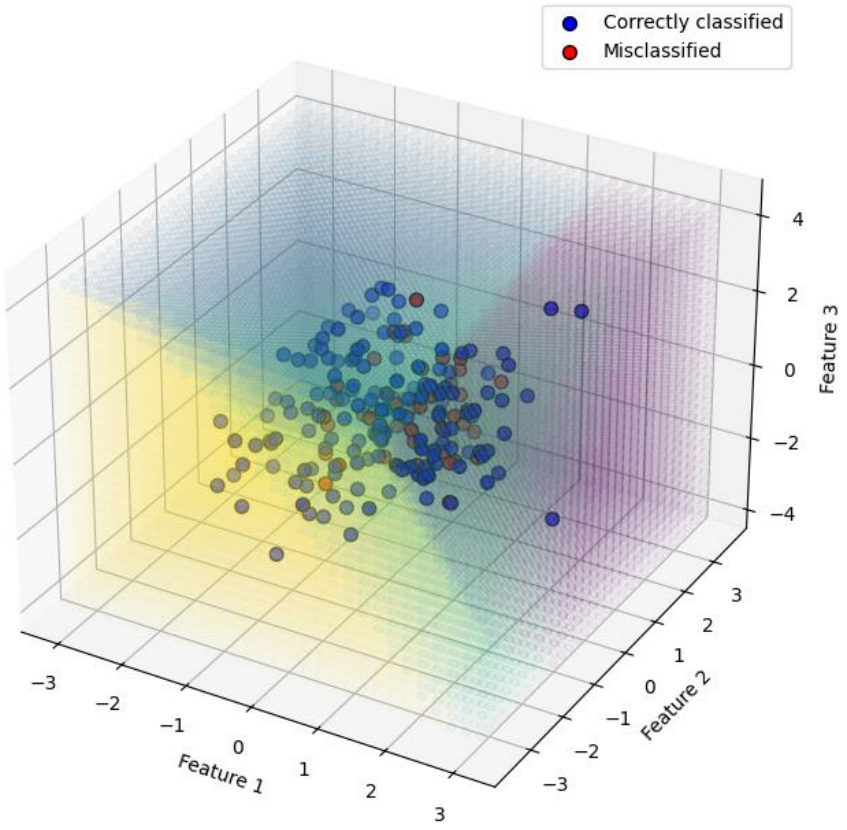


نمایش این نمودار به صورت interactive در داخل فایل Q1.1.ipynb قابل رویت است.

در ادامه نقاطی که درست پیش‌بینی شده اند را نسبت به نقاطی که اشتباه پیش‌بینی شده‌اند نمایش داده شده است.

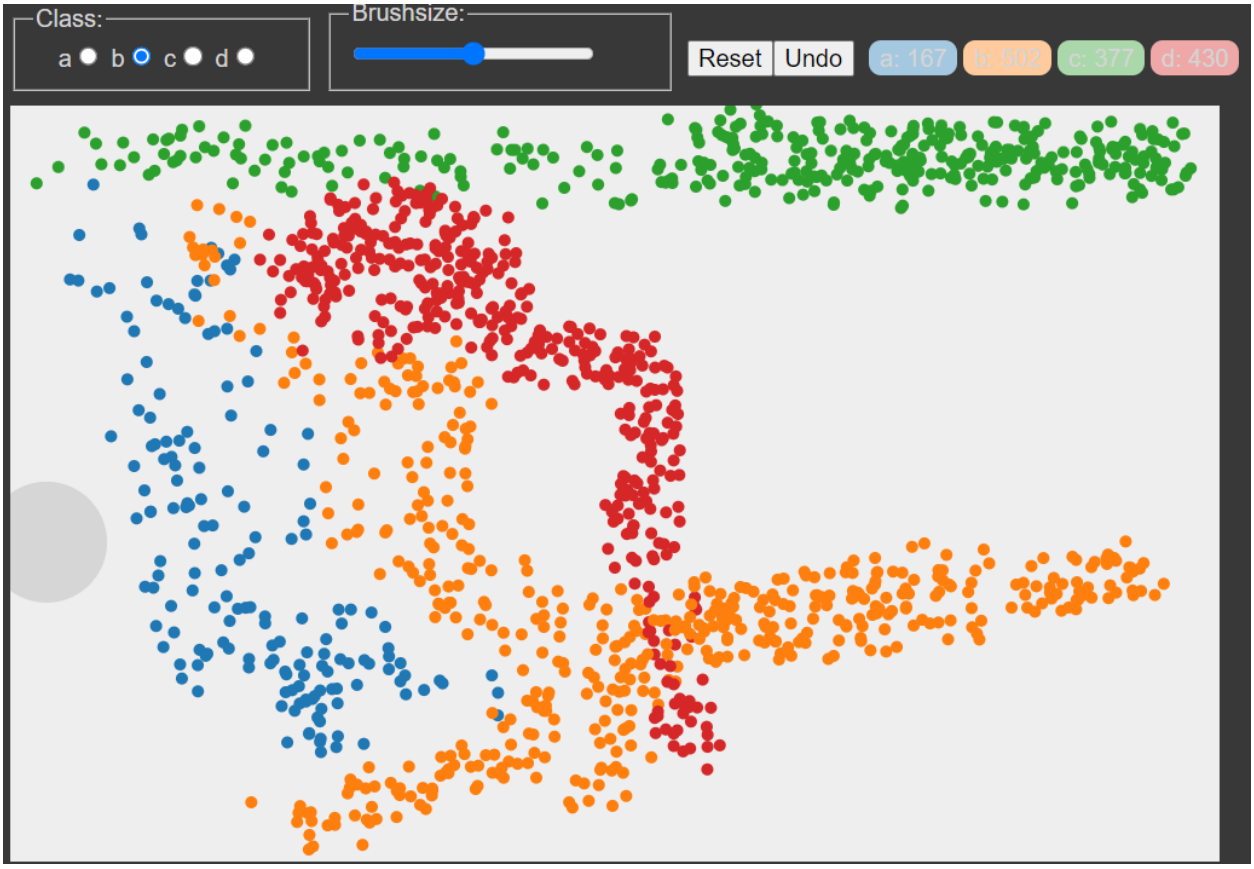


SGDClassifier, penalty=l2, loss='log_loss', alpha=1e-6, max_iter=4000, tol=1e-6, learning_rate='adaptive'
, eta0=2.632526, early_stopping=True, validation_fraction=i, n_iter_no_change=10, random_state=53



۵. فرآیندی مشابه قسمت «۲» را با تعداد کلاس و ویژگی دلخواه؛ اما با استفاده از ابزار drawdata تکرار کنید . قسمت های «۳» و «۴» را برای این داده های جدید تکرار و نتایج را به صورتی مناسب نشان دهید.

در ابتدا مجموعه داده را با استفاده از کتابخانه drawdata ایجاد می‌کنیم.



مجموعه داده فوق به منظور آموزش و ارزیابی ساختار یادگیری ماشین استفاده شده است. ۲۰ درصد از این مجموعه داده برای ارزیابی و ۸۰ درصد آن برای آموزش استفاده شده است.

تمام مراحلی که در قسمت قبل توضیح داده شد را روی این مجموعه داده پیاده می‌کنیم که تصاویر نتایج بخش‌های گوناگون به ترتیب زیر می‌باشد.

```
These results are for training the LogisticRegression model before apply the normalization transform on the training and test set.
model with sag solver and multi_class=multinomial score on train set is equal ot 0.5440677966101695
model with sag solver and multi_class=multinomial score on test set is equal ot 0.5675675675675675
test set: accuracy = 0.5675675675675675, recall = 0.4689517233821031, precision = 0.416779928723059

model with saga solver and multi_class=multinomial score on train set is equal ot 0.5432203389830509
model with saga solver and multi_class=multinomial score on test set is equal ot 0.5675675675675675
test set: accuracy = 0.5675675675675675, recall = 0.4689517233821031, precision = 0.41862960137325955

model with lbfgs solver and multi_class=multinomial score on train set is equal ot 0.8491525423728814
model with lbfgs solver and multi_class=multinomial score on test set is equal ot 0.8581081081081081
test set: accuracy = 0.8581081081081081, recall = 0.8493506493506493, precision = 0.8628756171970826

model with newton-cg solver and multi_class=multinomial score on train set is equal ot 0.8525423728813559
model with newton-cg solver and multi_class=multinomial score on test set is equal ot 0.8581081081081081
test set: accuracy = 0.8581081081081081, recall = 0.8493506493506493, precision = 0.8642871136922619


  ------------------------------------------------------------
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
model with sag solver and multi_class=ovr score on train set is equal ot 0.5627118644067797
model with sag solver and multi_class=ovr score on test set is equal ot 0.5709459459459459
test set: accuracy = 0.5709459459459459, recall = 0.4686530768092495, precision = 0.3903381753892235

model with saga solver and multi_class=ovr score on train set is equal ot 0.4762711864406795
model with saga solver and multi_class=ovr score on test set is equal ot 0.5067567567567568
test set: accuracy = 0.5067567567567568, recall = 0.4089429557839884, precision = 0.2927389705823534

model with lbfgs solver and multi_class=ovr score on train set is equal ot 0.8161016949152542
model with lbfgs solver and multi_class=ovr score on test set is equal ot 0.8344594594594594
test set: accuracy = 0.8344594594594594, recall = 0.8071428571428572, precision = 0.8464960858097313

model with liblinear solver and multi_class=ovr score on train set is equal ot 0.7703389830508475
model with liblinear solver and multi_class=ovr score on test set is equal ot 0.7837837837837838
test set: accuracy = 0.7837837837837838, recall = 0.7389993972272454, precision = 0.8293821749495691

model with newton-cg solver and multi_class=ovr score on train set is equal ot 0.8161016949152542
model with newton-cg solver and multi_class=ovr score on test set is equal ot 0.8344594594594594
test set: accuracy = 0.8344594594594594, recall = 0.8071428571428572, precision = 0.8464960858097313

model with newton-cholesky solver and multi_class=ovr score on train set is equal ot 0.8161016949152542
model with newton-cholesky solver and multi_class=ovr score on test set is equal ot 0.8344594594594594
test set: accuracy = 0.8344594594594594, recall = 0.8071428571428572, precision = 0.8464960858097313

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

همانطور که مشخص است پیش از اعمال normalization مدل‌ها همگرا نشده‌اند که این به معنی بهینه نبودن فرایند آموزش و عملکرد مدل‌ها است. در ادامه مدل روی داده‌های نرمال شده آموزش‌داده‌شده است.

```
model with sag solver and multi_class=ovr score on train set is equal to 0.7966101694915254
model with sag solver and multi_class=ovr score on test set is equal to 0.8209459459459459
test set: accuracy = 0.8209459459459459, recall = 0.7841772151898735, precision = 0.8564135806515042

model with saga solver and multi_class=ovr score on train set is equal to 0.7966101694915254
model with saga solver and multi_class=ovr score on test set is equal to 0.8209459459459459
test set: accuracy = 0.8209459459459459, recall = 0.7841772151898735, precision = 0.8564135806515042

model with lbfgs solver and multi_class=ovr score on train set is equal to 0.7966101694915254
model with lbfgs solver and multi_class=ovr score on test set is equal to 0.8209459459459459
test set: accuracy = 0.8209459459459459, recall = 0.7841772151898735, precision = 0.8564135806515042

model with liblinear solver and multi_class=ovr score on train set is equal to 0.7864406779661017
model with liblinear solver and multi_class=ovr score on test set is equal to 0.8040540540540541
test set: accuracy = 0.8040540540540541, recall = 0.7642939339141871, precision = 0.8415916819074714

model with newton-cg solver and multi_class=ovr score on train set is equal to 0.7966101694915254
model with newton-cg solver and multi_class=ovr score on test set is equal to 0.8209459459459459
test set: accuracy = 0.8209459459459459, recall = 0.7841772151898735, precision = 0.8564135806515042

model with newton-cholesky solver and multi_class=ovr score on train set is equal to 0.7966101694915254
model with newton-cholesky solver and multi_class=ovr score on test set is equal to 0.8209459459459459
test set: accuracy = 0.8209459459459459, recall = 0.7841772151898735, precision = 0.8564135806515042
```

```
These results are obtained from training LogisticRegression model with different solvers and different multi_class arguments

model with sag solver and multi_class=multinomial score on train set is equal to 0.8432203389830508
model with sag solver and multi_class=multinomial score on test set is equal to 0.8547297297297297
test set: accuracy = 0.8547297297297297, recall = 0.8422077922077922, precision = 0.8599756579362824

model with saga solver and multi_class=multinomial score on train set is equal to 0.8432203389830508
model with saga solver and multi_class=multinomial score on test set is equal to 0.8547297297297297
test set: accuracy = 0.8547297297297297, recall = 0.8422077922077922, precision = 0.8599756579362824

model with lbfgs solver and multi_class=multinomial score on train set is equal to 0.8432203389830508
model with lbfgs solver and multi_class=multinomial score on test set is equal to 0.8547297297297297
test set: accuracy = 0.8547297297297297, recall = 0.8422077922077922, precision = 0.8599756579362824

model with newton-cg solver and multi_class=multinomial score on train set is equal to 0.8432203389830508
model with newton-cg solver and multi_class=multinomial score on test set is equal to 0.8547297297297297
test set: accuracy = 0.8547297297297297, recall = 0.8422077922077922, precision = 0.8599756579362824

------------------------------------------------------------
```

همانطور که مشخص است نتایج روی داده‌های نرمال بهتر است. حال به محاسبه فراپارامتر C می‌پردازیم.

```
These results indicate the beset value of C between 0.01 and 1.5

 model with sag solver and multi_class=multinomial score on train set is equal to 0.8432203389830508. C=0.81
 model with sag solver and multi_class=multinomial score on test set is equal to 0.8547297297297297
 test set: accuracy = 0.8547297297297297, recall = 0.8422077922077922, precision = 0.8599756579362824

 model with saga solver and multi_class=multinomial score on train set is equal to 0.8432203389830508. C=0.81
 model with saga solver and multi_class=multinomial score on test set is equal to 0.8547297297297297
 test set: accuracy = 0.8547297297297297, recall = 0.8422077922077922, precision = 0.8599756579362824

 model with lbfgs solver and multi_class=multinomial score on train set is equal to 0.8432203389830508. C=0.8
 model with lbfgs solver and multi_class=multinomial score on test set is equal to 0.8547297297297297
 test set: accuracy = 0.8547297297297297, recall = 0.8422077922077922, precision = 0.8599756579362824

 model with newton-cg solver and multi_class=multinomial score on train set is equal to 0.8432203389830508. C=0.8
 model with newton-cg solver and multi_class=multinomial score on test set is equal to 0.8547297297297297
 test set: accuracy = 0.8547297297297297, recall = 0.8422077922077922, precision = 0.8599756579362824
```

```
model with sag solver and multi_class=ovr score on train set is equal to 0.8008474576271186. C=1.33
model with sag solver and multi_class=ovr score on test set is equal to 0.8277027027027027
test set: accuracy = 0.8277027027027027, recall = 0.791289933694997, precision = 0.85384419803748

model with saga solver and multi_class=ovr score on train set is equal to 0.8008474576271186. C=1.34
model with saga solver and multi_class=ovr score on test set is equal to 0.8277027027027027
test set: accuracy = 0.8277027027027027, recall = 0.791289933694997, precision = 0.85384419803748

model with lbfgs solver and multi_class=ovr score on train set is equal to 0.8008474576271186. C=1.34
model with lbfgs solver and multi_class=ovr score on test set is equal to 0.8277027027027027
test set: accuracy = 0.8277027027027027, recall = 0.791289933694997, precision = 0.85384419803748

model with liblinear solver and multi_class=ovr score on train set is equal to 0.7932203389830509. C=1.48
model with liblinear solver and multi_class=ovr score on test set is equal to 0.8175675675675675
test set: accuracy = 0.8175675675675675, recall = 0.7770343580470163, precision = 0.8533271015376278

model with newton-cg solver and multi_class=ovr score on train set is equal to 0.8008474576271186. C=1.34
model with newton-cg solver and multi_class=ovr score on test set is equal to 0.8277027027027027
test set: accuracy = 0.8277027027027027, recall = 0.791289933694997, precision = 0.85384419803748

model with newton-cholesky solver and multi_class=ovr score on train set is equal to 0.8008474576271186. C=1.34
model with newton-cholesky solver and multi_class=ovr score on test set is equal to 0.8277027027027027
test set: accuracy = 0.8277027027027027, recall = 0.791289933694997, precision = 0.85384419803748
```

در گام بعدی دو مدل که عملکرد بهتری دارند را نگه می‌داریم و بهترین میزان tol را برایشان محاسبه می‌کنیم.

```
among different solvers and multi_class approaches the newton-cg solver with multinomial approach and liblinear solver with ovr selected and the process will be continued

model with newton solver and multi_class=multinomial score on train set is equal to 0.8432203389830508. tol=0.8
model with newton solver and multi_class=multinomial score on test set is equal to 0.8547297297297297
test set: accuracy = 0.8547297297297297, recall = 0.8422077922077922, precision = 0.8599756579362824

model with liblinear solver and multi_class=ovr score on train set is equal to 0.7932203389830509. tol=1e-06
model with liblinear solver and multi_class=ovr score on test set is equal to 0.8175675675675675
test set: accuracy = 0.8175675675675675, recall = 0.7770343580470163, precision = 0.8533271015376278
```

در انتها فراپارامتر dual را برای liblinear فعال می‌کنیم و مدل را آموزش می‌دهیم.

```
The dual parameter is only applicable for liblinear solver thus the solver is liblinear and dual is true.
model with liblinear solver and multi_class=ovr score on train set is equal to 0.7932203389830509.
model with liblinear solver and multi_class=ovr score on test set is equal to 0.8175675675675675
test set: accuracy = 0.8175675675675675, recall = 0.7770343580470163, precision = 0.8533271015376278
```

بهترین مدل LogisticRegresion در آخرین آموزش حاصل شده است. حال به سراغ مدل SGDClassifier می‌رویم.

```
the SGDClassifier with different losses is calculated here

model with hinge loss function score is equal ot 0.7728813559322034 on the train set
model with hinge loss function score is equal ot 0.7871621621621622 on the test set
test set: accuracy = 0.7871621621621622, recall = 0.7569264069264069, precision = 0.8609801224743754

model with log_loss loss function score is equal ot 0.8144067796610169 on the train set
model with log_loss loss function score is equal ot 0.8547297297297297 on the test set
test set: accuracy = 0.8547297297297297, recall = 0.8361471861471862, precision = 0.8676587538623066

model with perceptron loss function score is equal ot 0.6025423728813559 on the train set
model with perceptron loss function score is equal ot 0.6385135135135135 on the test set
test set: accuracy = 0.6385135135135135, recall = 0.6587402049427366, precision = 0.541935050993022

model with squared_error loss function score is equal ot 0.1669491525423729 on the train set
model with squared_error loss function score is equal ot 0.13513513513513514 on the test set
test set: accuracy = 0.13513513513513514, recall = 0.15714285714285714, precision = 0.06695557963163597
```

تصویر فوق نشان می‌دهد که این مدل به روی دیتای نرمال شده چگونه عمل می‌کند.

```
Models with better performance selected and remains and the learning_rate is adaptive

model with hinge loss function score is equal ot 0.6440677966101694 on the train set
model with hinge loss function score is equal ot 0.668918918918919 on the test set
test set: accuracy = 0.668918918918919, recall = 0.5541509123787605, precision = 0.49074790624086395

model with log_loss loss function score is equal ot 0.6720338983050848 on the train set
model with log_loss loss function score is equal ot 0.7162162162162162 on the test set
test set: accuracy = 0.7162162162162162, recall = 0.6004712586991068, precision = 0.547187217976129

model with perceptron loss function score is equal ot 0.8101694915254237 on the train set
model with perceptron loss function score is equal ot 0.7972972972972973 on the test set
test set: accuracy = 0.7972972972972973, recall = 0.767532467532467, precision = 0.810179255129496

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

حال مقدار eta0 بهینه را مشخص می‌کنیم. همان‌طور که مشخص است عملکرد بعضی مدل‌ها کاهش پیدا کرده؛ اما این به دلیل مقدار eta0 اولیه است و با اصلاح آن نتیجه بهتر خواهد شد؛ بنابراین به سراغ بهینه‌سازی مقدار eta0 می‌رویم.

```
The best eta0 is selected between 1e13 and 20
model with hinge loss function score is equal ot 0.8152542372881356 on the train set. eta0=5.264052631578948
model with hinge loss function score is equal ot 0.8344594594594594 on the test set. eta0=5.264052631578948
test set: accuracy = 0.8344594594594594, recall = 0.8218614718614718, precision = 0.8486149149097423

model with log_loss loss function score is equal ot 0.811864406779661 on the train set. eta0=34.210842105263154
model with log_loss loss function score is equal ot 0.8277027027027027 on the test set. eta0=34.210842105263154
test set: accuracy = 0.8277027027027027, recall = 0.8023809523809524, precision = 0.8396593728514632

model with perceptron loss function score is equal ot 0.8313559322033899 on the train set. eta0=13.158631578947368
model with perceptron loss function score is equal ot 0.8445945945945946 on the test set. eta0=13.158631578947368
test set: accuracy = 0.8445945945945946, recall = 0.8389610389610389, precision = 0.8472479332694387
```

با بهینه‌سازی مقدار eta0 و بررسی عملکرد مدل‌ها به این نتیجه می‌رسیم که adaptive learning rate می‌تواند به بهبود نتایج کمک کند. حال دو مدلی که عملکرد بهتری دارند را انتخاب کرده و باقی آموزش‌ها را به روی آنها انجام می‌دهیم.

```
model with hinge loss function score is equal ot 0.8152542372881356 on the train set. tol=1e-06
model with hinge loss function score is equal ot 0.8344594594594594 on the test set. tol=1e-06
test set: accuracy = 0.8344594594594594, recall = 0.8218614718614718, precision = 0.8486149149097423

model with perceptron loss function score is equal ot 0.8144067796610169 on the train set. tol=1e-06
model with perceptron loss function score is equal ot 0.847972972972973 on the test set. tol=1e-06
test set: accuracy = 0.847972972972973, recall = 0.830952380952381, precision = 0.8562195119753627
```

در ادامه بهترین مقدار فراپارامتر alpha را حساب می‌کنیم.

```
The best value for alpha obtained between 1e-6 and 1e-2

model with hinge loss function score is equal ot 0.809322033898305 on the train set. alpha=1e-06
model with hinge loss function score is equal ot 0.831081081081081 on the test set. aplha=1e-06
test set: accuracy = 0.831081081081081, recall = 0.8108225108225109, precision = 0.8446446292052674

model with perceptron loss function score is equal ot 0.8440677966101695 on the train set. alpha=0.008947473684210526
model with perceptron loss function score is equal ot 0.8783783783783784 on the test set. aplha=0.008947473684210526
test set: accuracy = 0.8783783783783784, recall = 0.8523809523809525, precision = 0.897303029461069
```

همان‌طور که مشخص است عملکرد مدل perceptreon اختلاف زیادی با مدل‌ها دیگر دارد.

در انتها بهترین نسبت مجموعه‌داده صحت‌سنجی نسبت به آموزش را به دست می‌آوریم.

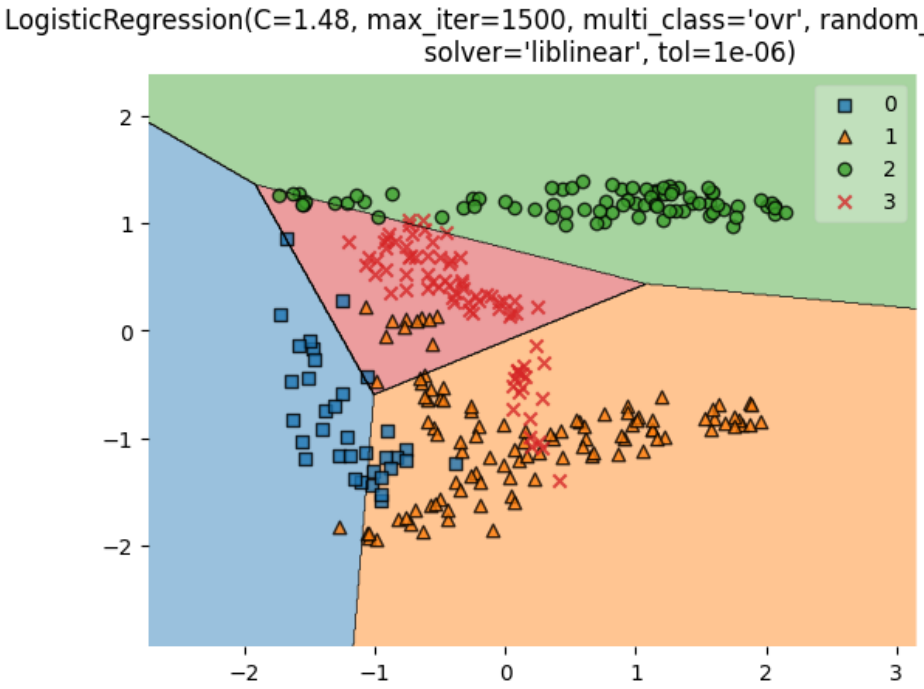
```
The best fraction ov train val is selected
model with hinge loss function score is equal ot 0.8110169491525424 on the train set. eta0=1e-06. validation_fraction=0.1
model with hinge loss function score is equal ot 0.8243243243243243 on the test set. eta0=1e-06
test set: accuracy = 0.8243243243243243, recall = 0.8043290043290043, precision = 0.8403431680267839

model with perceptron loss function score is equal ot 0.8440677966101695 on the train set. eta0=0.008947473684210526. validation_fraction=0.1
model with perceptron loss function score is equal ot 0.8783783783783784 on the test set. eta0=0.008947473684210526
test set: accuracy = 0.8783783783783784, recall = 0.8523809523809525, precision = 0.897303029461069
```

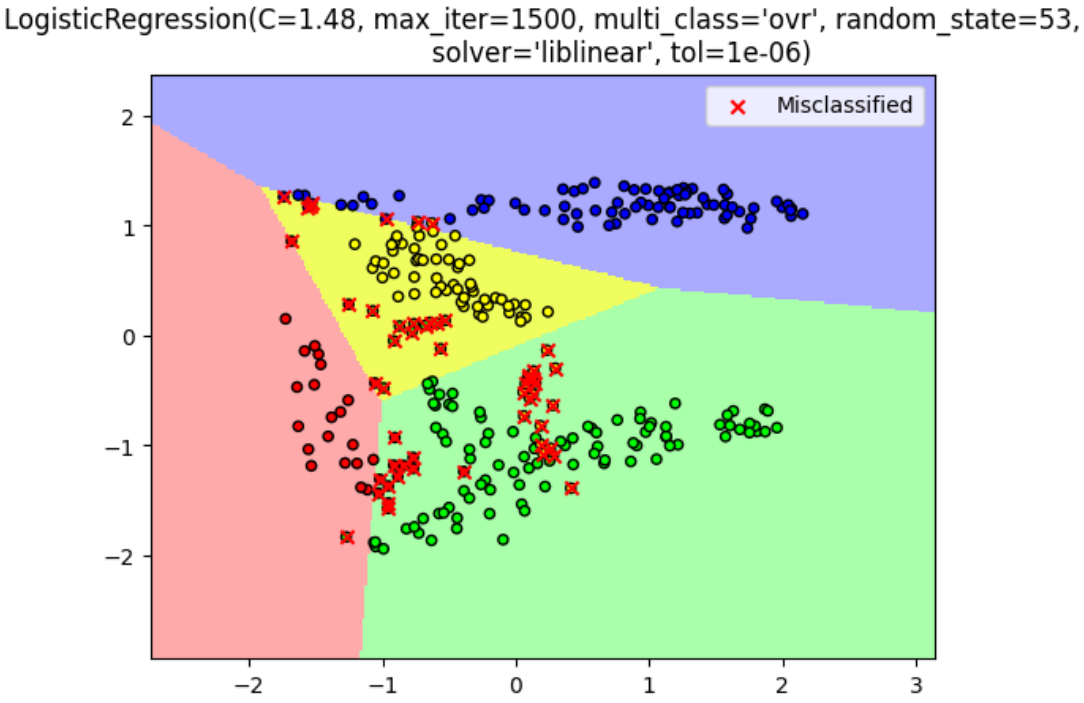
همان‌طور که از مقایسه قابل‌تشخیص است بهترین مدل SGDClassifier نسبت به بهترین مدل LogisticRegression عملکرد مطلوب‌تری دارد؛ بنابراین اگر یک مدل با مشخصات زیر آموزش ببیند می‌تواند بهترین نتیجه را در پیش‌بینی کلاس‌های این دیتاست داشته باشد.

```
SGDClassifier(penalty="l2",
                      loss='perceptron',
                      alpha=0.008947473684210526,
                      max_iter=4000,
                      tol=1e-6,
```

```
                    learning_rate='adaptive',
                    eta0=13.15863157,
                    early_stopping=True,
                    validation_fraction=0.1,
                    n_iter_no_change=10,
                    random_state=53)
```

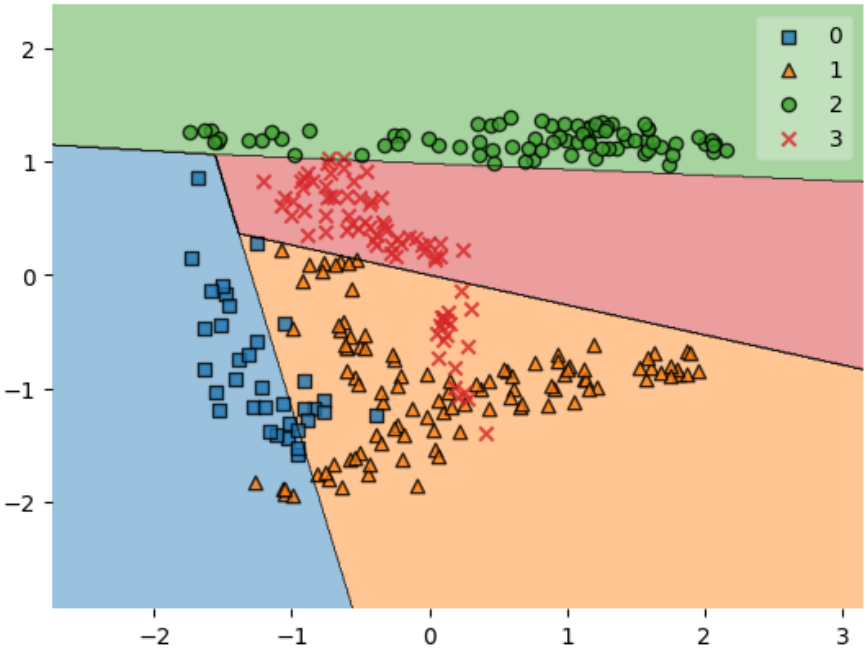
حال به‌رسم نمودارهای خواسته شده در صورت سؤال، برای مدل LogisticRegression می‌پردازیم.



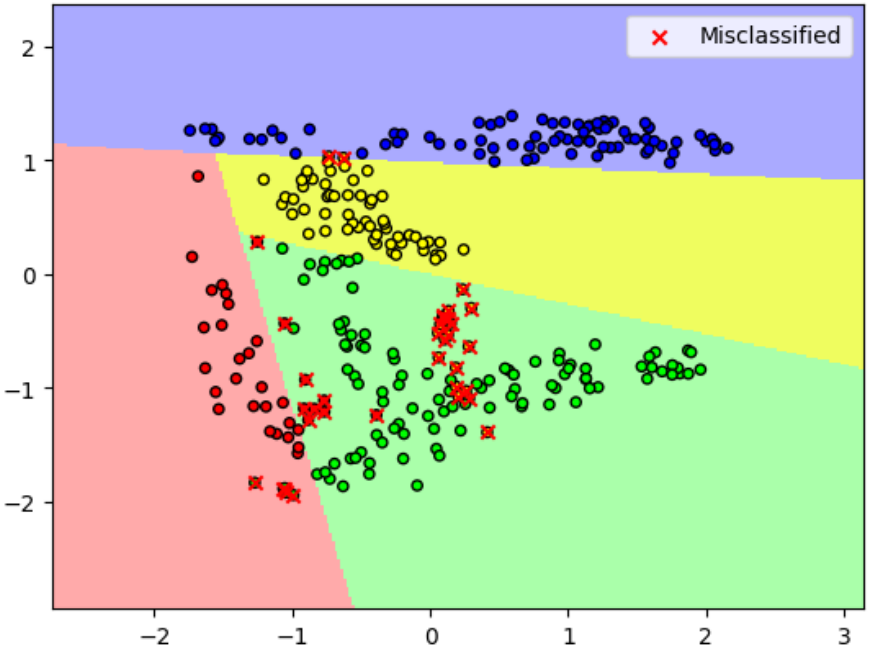


حال نتایج را برای مدل SGDClassifier بررسی می‌کنیم.

SGDClassifier(alpha=0.008947473684210526, early_stopping=True, eta0=13.15863157,
learning_rate='adaptive', loss='perceptron', max_iter=4000,
n_iter_no_change=10, random_state=53, tol=1e-06)



SGDClassifier(alpha=0.008947473684210526, early_stopping=True, eta0=13.15863157,
learning_rate='adaptive', loss='perceptron', max_iter=4000,
n_iter_no_change=10, random_state=53, tol=1e-06)

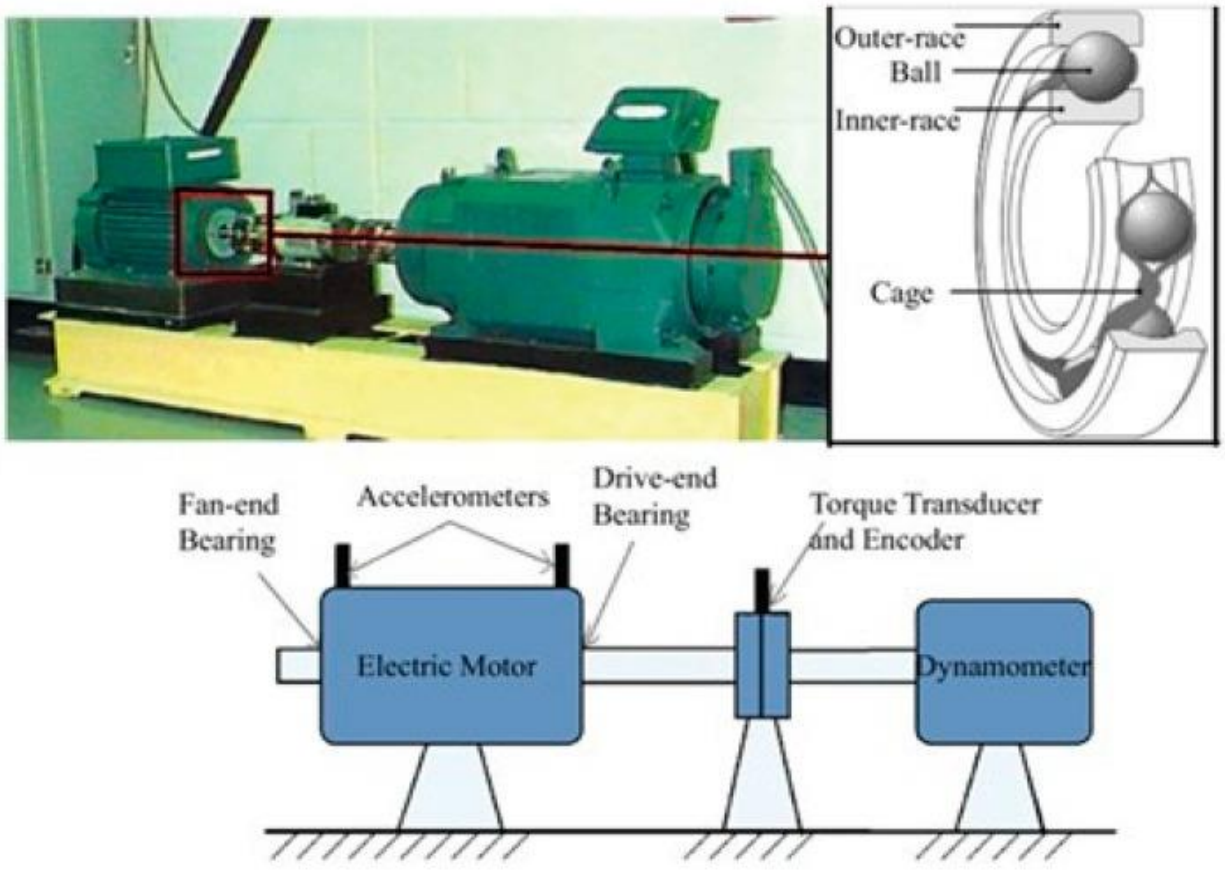


سؤال ۲

۱. با مراجعه به صفحهٔ دیتاست Bearing CWRU با یک دیتاست مربوط به حوزهٔ «تشخیص عیب» آشنا شوید. با جستجوی آن در اینترنت و مقالات، توضیحاتی از اهداف، ویژگی ها و حالت های مختلف این دیتاست ارائه کنید .در ادامه، ابتدا به صفحهٔ داده های سالم مراجعه کنید و داده های کلاس سالم(X_Normal) ۱) را دریافت کنید. سپس، به صفحهٔ داده های عیب در حالت k12 مراجعه کرده و داده های کلاس عیب) X_007IR) را دریافت کنید.

CWRU Bearing به عنوان یک مجموعه داده جامع و معیار (Benchmark) برای ارزیابی الگوریتم‌های یادگیری ماشین با وظیفه (Task) طبقه‌بندی (Classification) در زمینه تشخیص عیب شناخته می‌شود. این مجموعه داده به منظور تشخیص خرابی در Ball Bearing توسعه یافته است. این مجموعه داده به منظور توسعه و آزمایش الگوریتم‌های تشخیص خرابی مورد استفاده قرار می‌گیرد.[i] [ii]

با توجه به شکل زیر ابزار موجود در جمع‌آوری دیتای CWRU، شامل یک موتور دو اسب بخار، token converter, encoder, dynamometer, و دستگاه کنترل است که به منظور جمع‌آوری داده به یکدیگر متصل شده اند. یک بلبرینگ معمولی شامل یک مسیر داخلی، یک مسیر خارجی، توپ ها و یک قفس است که توپ ها را در جای خود نگه می دارد. بلبرینگ شفت موتور گشتاور را از طریق dynamometer و سیستم کنترل الکترونیکی به شافت منتقل می شود[iii] .



این مجموعه شامل سیگنال‌های لرزش ball bearingها است. سنسورهای لرزش در فرایند جمع‌آوری دیتا در انتهای drive، fan و Basement قرار داده شده است. در ادامه به منظور ایجاد شرایط نزدیک به حالت واقع، عیوب به صورت مصنوعی در دستگاه ایجاد می‌شوند

تا توسط حسگرها مقدار لرزش دستگاه معیوب حین کار اندازه‌گیری شود. به صورت کلی با استفاده از داده‌های حسگرها می‌توان سه وضعیت عادی، عیب در drive و عیب در fan را مشخص کرد.[iii]
حسگرها با نرخ ۱۲۰۰۰ نمونه در ثانیه از سه محل متفاوت میزان لرزش را ثبت کرده‌اند این درحالی است که برای حسگری که در محل drive قرار دارد با نرخ ۴۸۰۰۰ نمونه در ثانیه نمونه‌برداری انجام شده است.[i]

۲. برای تشکیل دیتاست مراحل زیر را انجام دهید

۱. ز هر کلاس M نمونه با طول N جدا کنید (M حداقل ۱۰۰ و N حداقل ۲۰۰ باشد .(یک ماتریس از داده های هر دو کلاس به همراه برچسب مربوطه تشکیل دهید. می توانید پنجره ای به طول N در نظر بگیرید و در نهایت یک ماتریس M × N از داده های هر کلاس استخراج کنید.

```
n_samples = 300
len_data = 200

normal_data_matrix = normal_data[:-(normal_data.shape[0] % len_data)].reshape(-1, len_data)
fault_data_matrix = fault_data[:-(fault_data.shape[0] % len_data)].reshape(-1, len_data)
print(normal_data_matrix.shape)
print(fault_data_matrix.shape)

normal_data_matrix = normal_data_matrix[:n_samples,:]
fault_data_matrix = fault_data_matrix[:n_samples,:]
print(normal_data_matrix.shape)
print(fault_data_matrix.shape)
```

```
(2419, 200)
(609, 200)
(300, 200)
(300, 200)
```

کد فوق به منظور حرکت یک پنجره با طول ۲۰۰ روی مجموعه داده می‌باشد. در نتیجه آن ماتریس داده‌های سالم شامل ۲۴۱۹ نمونه و ماتریس داده‌های ناسالم شامل ۶۰۹ نمونه است. در ادامه تعداد ۳۰۰ نمونه از هرکدام جدا شده تا در فرایند آموزش و ارزیابی استفاده شود.

۲. در مورد اهمیت استخراج ویژگی در یادگیری ماشین توضیحاتی بنویسید. سپس، با استفاده از حداقل ۸ عدد از روش های ذکرشده در جدول ۱، ویژگی های دیتاست قسمت «۲-آ» را استخراج کنید و یک دیتاست جدید تشکیل دهید.

استخراج ویژگی در الگوریتم‌های یادگیری ماشین خصوصاً الگوریتم‌های قدیمی‌تر از مهم‌ترین قسمت‌ها است. روش‌های استخراج ویژگی مبتنی بر روش‌های گوناگون ریاضی هستند و متناسب با وظیفه ساختار یادگیری ماشین و نوع داده‌های ورودی باید از بین آنها بهترین‌ها را انتخاب کرد. به‌عنوان‌مثال تبدیل فوریه یکی از انواع ویژگی است که می‌توان آن را از داده‌هایی که خاصیت فرکانسی دارند استخراج کرد و تنها هارمونیک‌هایی که شامل اطلاعات موردنیاز مسئله هست را بررسی کرد. گروهی دیگر از روش‌های استخراج ویژگی مقادیر نسبی را به‌عنوان خروجی خود نشان می‌دهند که با استفاده از این گروه از روش‌ها می‌توانیم حساسیت ساختار یادگیری ماشین را نسبت به تغییرات عملکرد حسگرها و یا تغییر دقت حسگرها کاهش دهیم که این مسئله باعث همگانی شدن ساختار می‌شود. از سوی دیگر گاهی اوقات ویژگی‌های مربوط به یک‌طبقه خاص در مسئله وابستگی به دامنه سیگنال‌های ورودی دارد که این مسئله با گروه دیگر از روش‌های استخراج ویژگی مانند RMS یا روش قله به دست می‌آیند. کاربرد دیگر روش‌های استخراج ویژگی در شرایطی است که ابعاد داده‌های ورودی زیاد است و امکان بررسی آنها به‌صورت مستقل وجود ندارد؛ بنابراین با روش‌های کاهش ابعاد مانند PCA مسئله را ساده‌تر می‌کنیم.

```python
normal_features = pd.DataFrame(feature_extraction(normal_data_matrix))
fault_features = pd.DataFrame(feature_extraction(fault_data_matrix))
normal_features['Label'] = 0
fault_features['Label'] = 1

selected_features = ['Standard Deviation', 'Peak', 'Skewness', 'Kurtosis',
                     'Crest Factor', 'Mean', 'Root Mean Square',
                     'Impulse Factor','Square Mean Root','Shape Factor','Label']

main_df = pd.concat([normal_features, fault_features], ignore_index=True)
df = main_df[selected_features]
df.head()
```

	Standard Deviation	Peak	Skewness	Kurtosis	Crest Factor	Mean	Root Mean Square	Impulse Factor	Square Mean Root	Shape Factor	Label
0	0.058989	0.198810	-0.728862	0.640523	3.368817	-0.001747	0.059015	4.365452	0.059015	1.295841	0
1	0.066826	0.184416	-0.466580	-0.272507	2.727056	0.010360	0.067625	3.265005	0.067625	1.197264	0
2	0.072411	0.174820	-0.366553	-0.483823	2.404593	0.006504	0.072702	2.947953	0.072702	1.225968	0
3	0.070446	0.197559	-0.348124	-0.255664	2.725963	0.017020	0.072473	3.266700	0.072473	1.198366	0
4	0.053036	0.152706	-0.472174	0.251543	2.879044	-0.000669	0.053041	3.692401	0.053041	1.282509	0

در ادامه برچسب‌های هرکدام از کلاس‌ها را در ستونی جدید به اسم label ذخیره می‌کنیم و ویژگی‌های موجود در متغیر selected_features را استخراج کرده یک مجموعه‌داده جدید را ایجاد می‌کنیم.

۳. اهمیت فرایند برزدن را مطرح کنید و مجموعه‌داده را به زیربخش آموزش و ارزیابی تقسیم کنید.

بر زدن در فرایند آموزش الگوریتم یادگیری ماشین باعث می‌شود که در گام نخست روش‌های مبتنی بر مشتق به‌صورت پایدارتر همگرا بشوند و احتمال وقوع یک batch با شرایطی خاص را کاهش می‌دهد. از سوی دیگر بر زدن باعث می‌شود که الگوریتم با احتمال بیشتری از نقاط بهینه محلی عبور کند و به سمت نقطه بهینه جهانی حرکت کند.

با استفاده از دستور زیر عملیات بر زدن مجموعه‌داده رخ خواهد داد.

```python
df_shuffled = df.sample(frac=1).reset_index(drop=True)
```

در ادامه با استفاده از دستور زیر مجموعه‌داده با نسبت ۰.۸ به ۰.۲ به زیرمجموعه‌های آموزش و ارزیابی تقسیم می‌شود این درحالی است که اب تنظیم پارامتر stratify از هر کلاس به تعداد مساوی انتخاب می‌کنیم تا با مشکل عدم تعادل روبرو نشویم.

```python
X_train_raw, X_test_raw, y_train, y_test = train_test_split(X, y,
test_size=0.20, stratify=y, random_state=53)
```

۴. حداقل دو روش برای نرمال سازی داده ها را با ذکر اهمیت این فرآیند توضیح دهید و با استفاده از یکی از این روش ها، داده ها را نرمال کنید. آیا از اطلاعات بخش «ارزیابی» در فرآیند نرمال سازی استفاده کردید؟ چرا؟

در ابتدا باید توضیح دهیم که فرایند نرمالیزه‌کردن در کدام ابعاد ماتریس داده انجام می‌شود. اگر تبدیل ریاضی در جهت بردار نمونه انجام شود به آن batch normalization می‌گویند و اگر در جهت بردار ویژگی استفاده شود به آن layer normalization می‌گویند.

روش اول نرمال کردن داده با استفاده از نرم دوم بردار داده است. در این روش با استفاده از فرمول زیر تمام مقادیر مربوط به یک ویژگی خاص که در بردار آن ویژگی قرار دارد نرمال می‌شود. این روش از نرمال کردن باعث می‌شود که علاوه بر حفظ جهت بردار داده، نسبت اهمیت هر درایه نسبت به باقی درایه‌ها ثابت بماند. این روش معمولا به صورت batch normalization استفاده می‌شود.

$$x_{norm,i} = \frac{x_i}{\sqrt{\Sigma\, x_j^2}} = \frac{x_i}{||X||}$$

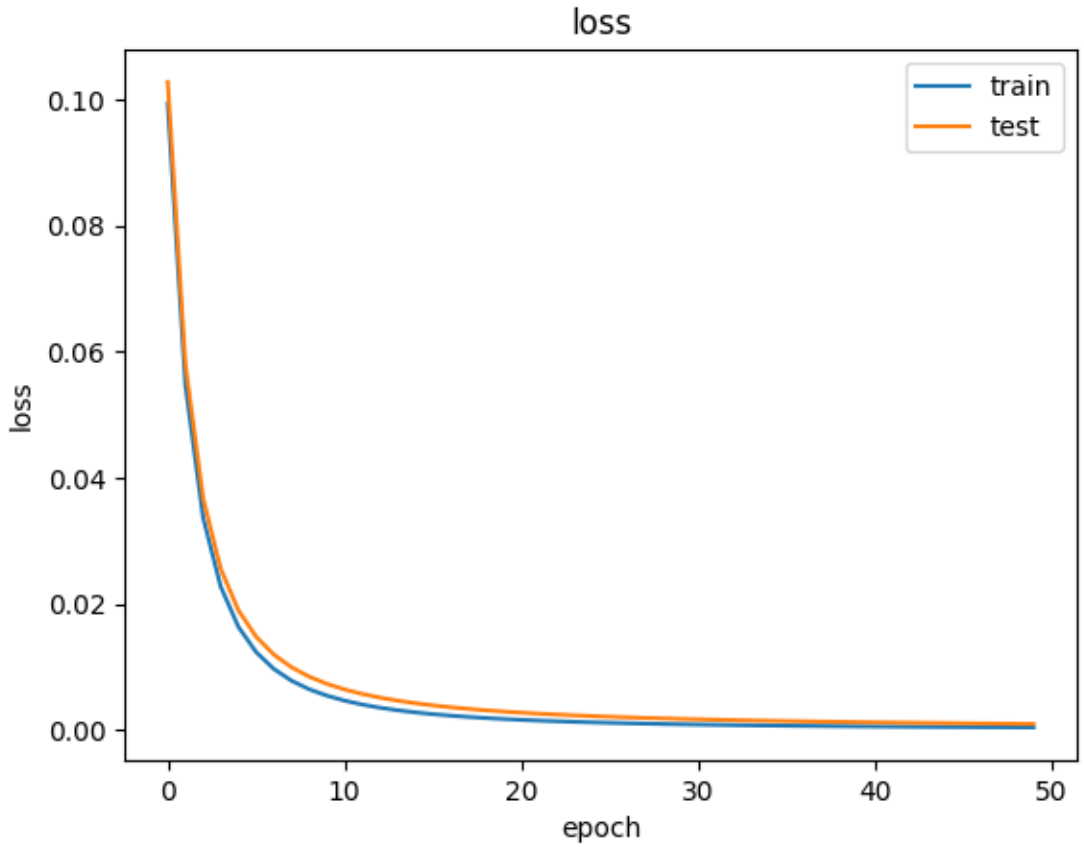
روش دوم نرمال کردن بر اساس حداکثر و حداقل مقدار است که این روش مقادیر را بین ۰ و ۱ تصویر می‌کند و در بسیاری از ساختارهای یادگیری ماشین خصوصا در پردازش تصاویر استفاده می‌شود. فرمول این روش به شکل زیر است.

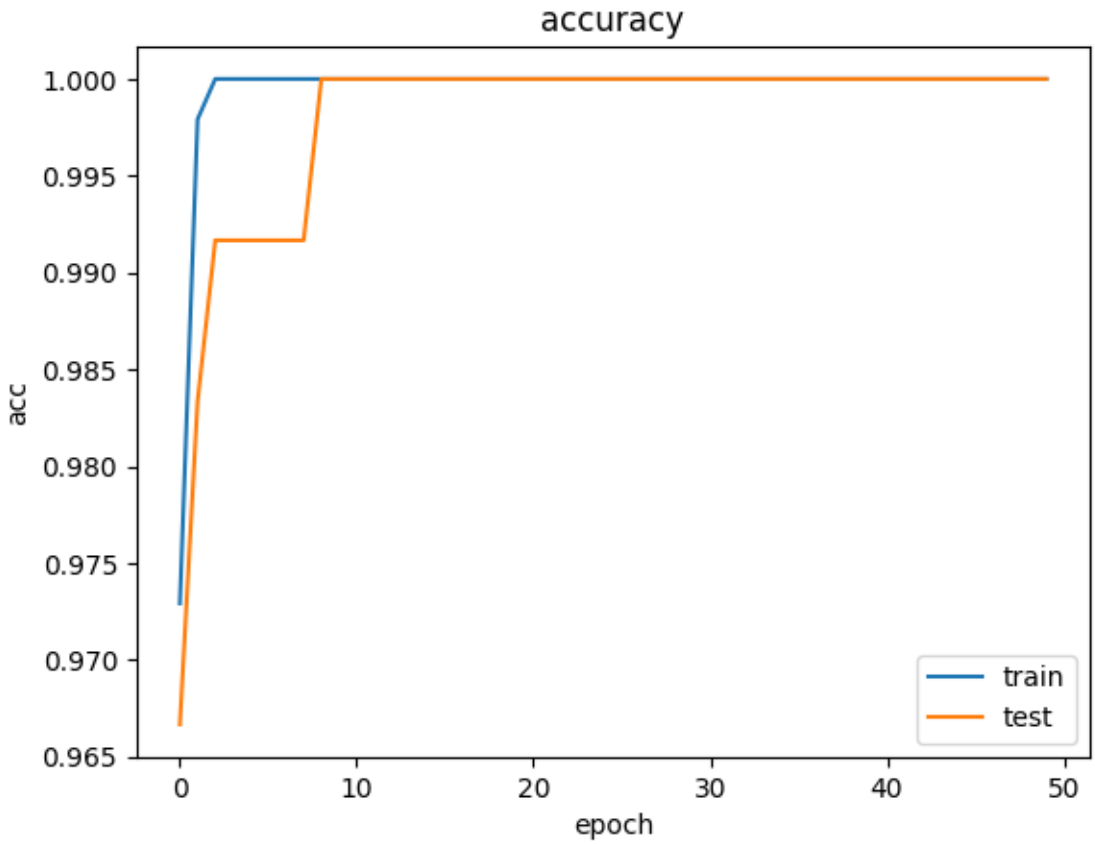
$$x_{norm,i} = \frac{x_i - \min(x)}{\max(X) - \min(X)}$$

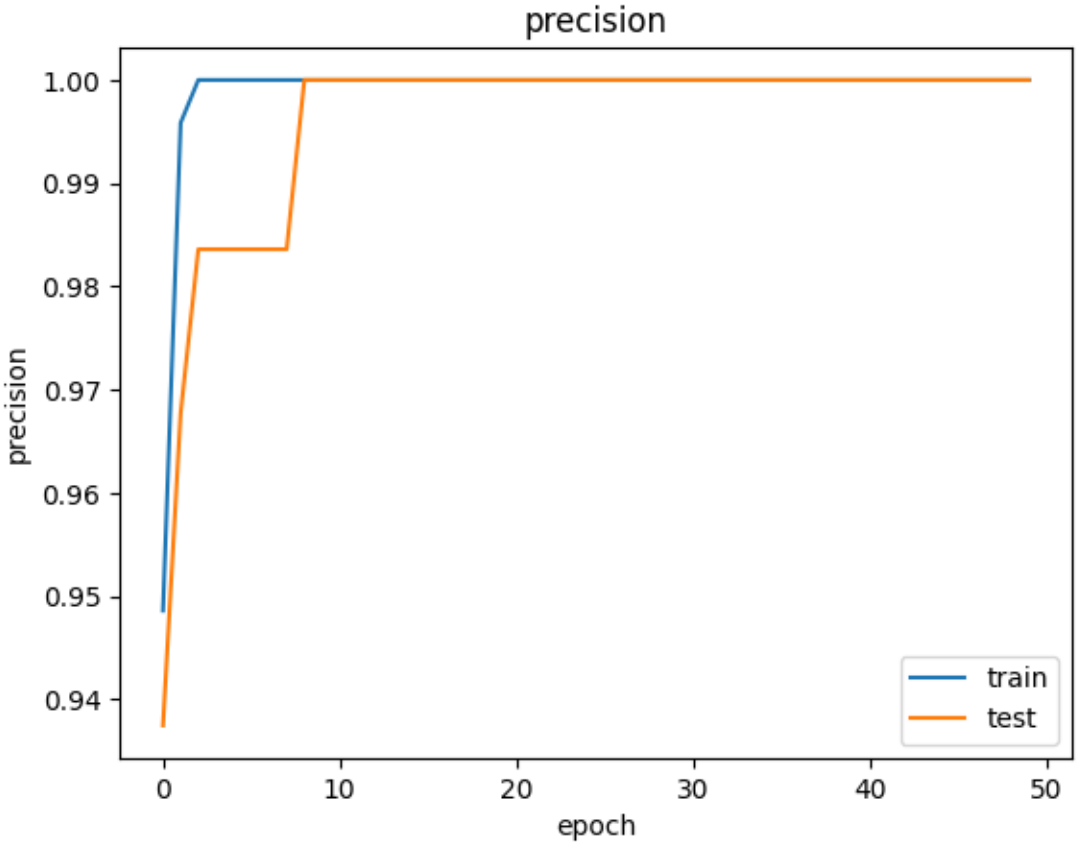
در این بخش از روش نرمال کردن حداکثر و حداقل استفاده شده است و محدوده داده‌ها به عددی بین ۰ و ۱ تصویر شده. نکته حائز اهمیت این است که از اطلاعات مجموعه داده ارزیابی در بدست آوردن مقادیر حداقل و حداکثر استفاده نشده است چون در شرایطی عملی داده‌های ارزیابی توسط ساختار یادگیری ماشین دیده نشده و ما از مقادیر آن اطلاعی نداریم بنابراین باید عملیات نرمال سازی را بر اساس داده‌های آموزش انجام دهیم و در ادامه هنگام ارزیابی مدل با استفاده از مقادیر بدست آمده از مجموعه داده آموزش، مجموعه داده ارزیابی را نرمال کنیم.

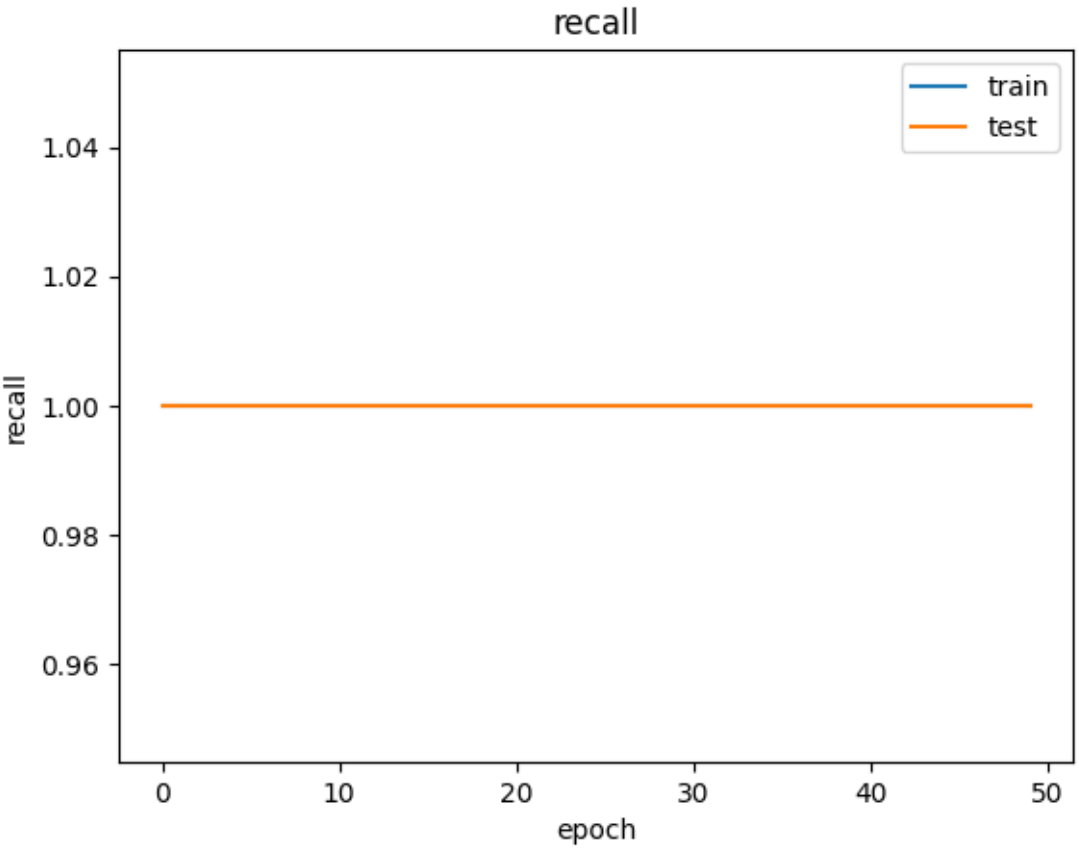
۳. بدون استفاده از کتابخانه های آمادهٔ پایتون، مدل طبقه بند، تابع اتلاف و الگوریتم یادگیری و ارزیابی را کدنویسی کنید تا دو کلاس موجود در دیتاست به خوبی از یکدیگر تفکیک شوند. نمودار تابع اتلاف را رسم کنید و نتیجهٔ ارزیابی روی داده های تست را با حداقل ۲ شاخصه محاسبه کنید. نمودار تابع اتلاف را تحلیل کنید. آیا می توان از روی نمودار تابع اتلاف و قبل از مرحلهٔ ارزیابی با قطعیت در مورد عمل کرد مدل نظر داد؟ چرا و اگر نمی توان، راه حل چیست؟

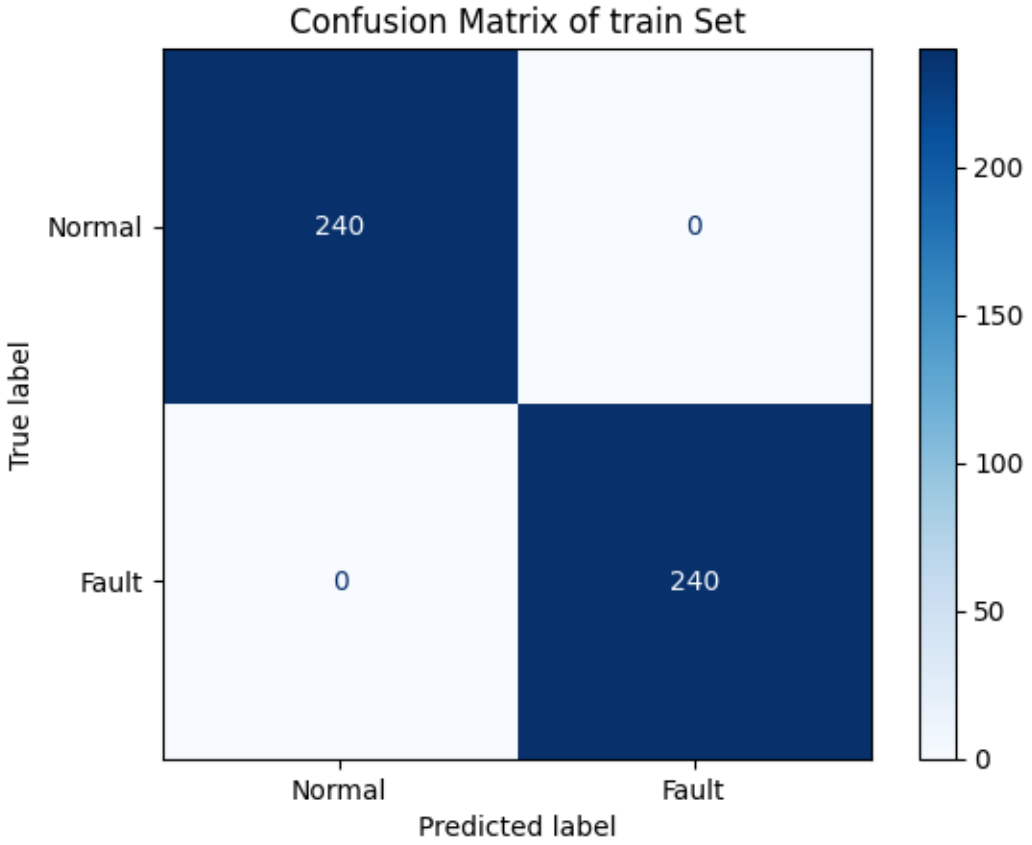
مدل یادگیری خطی Perceptron به‌منظور پیاده‌سازی انتخاب شد و با کمک ChatGPT کد آن نوشته شده است. این مدل با نرخ یادگیری ۰.۰۱، تابع هزینه MSE و کاهش نرخ یادگیری به ازای هر ۱۰ epoch بدون بهبود در هزینه تابع ارزیابی آموزش دیده است.
نمودارهای مربوط به این آموزش و ارزیابی‌های آن به شکل زیر است.

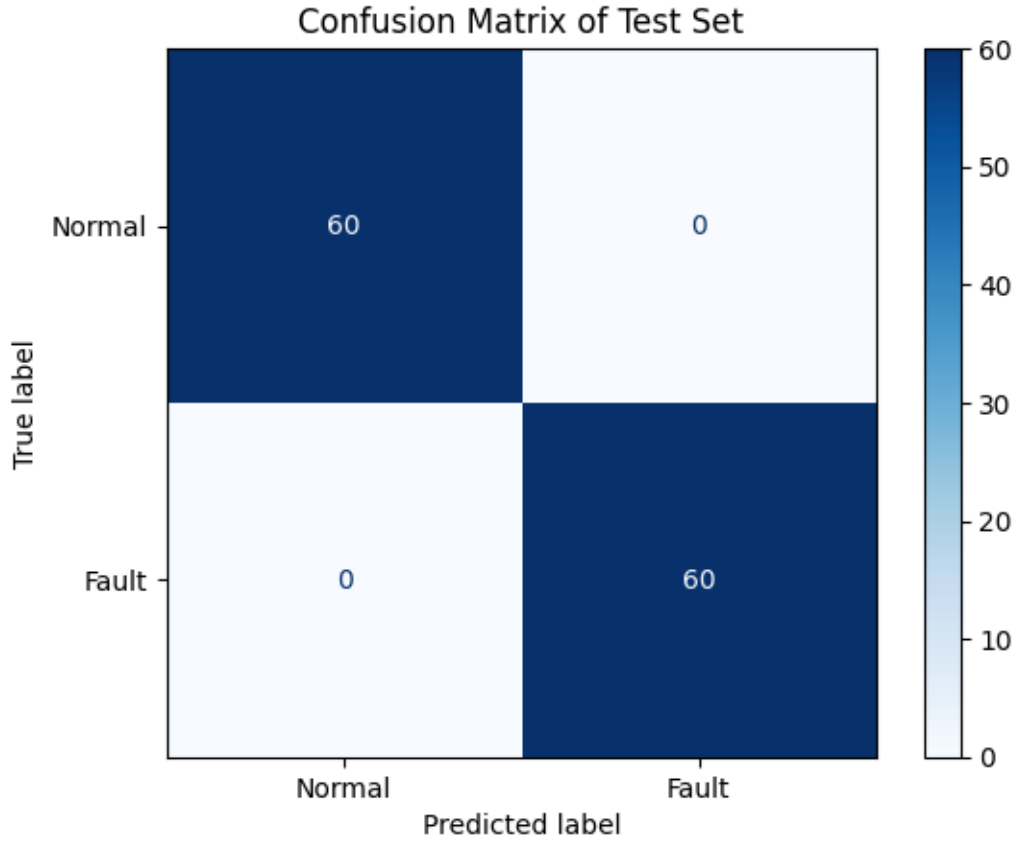



accuracy
1.000
0.995
0.990
0.985
0.980
0.975
0.970
0.965
acc
0
10
20
30
40
50
epoch
train
test

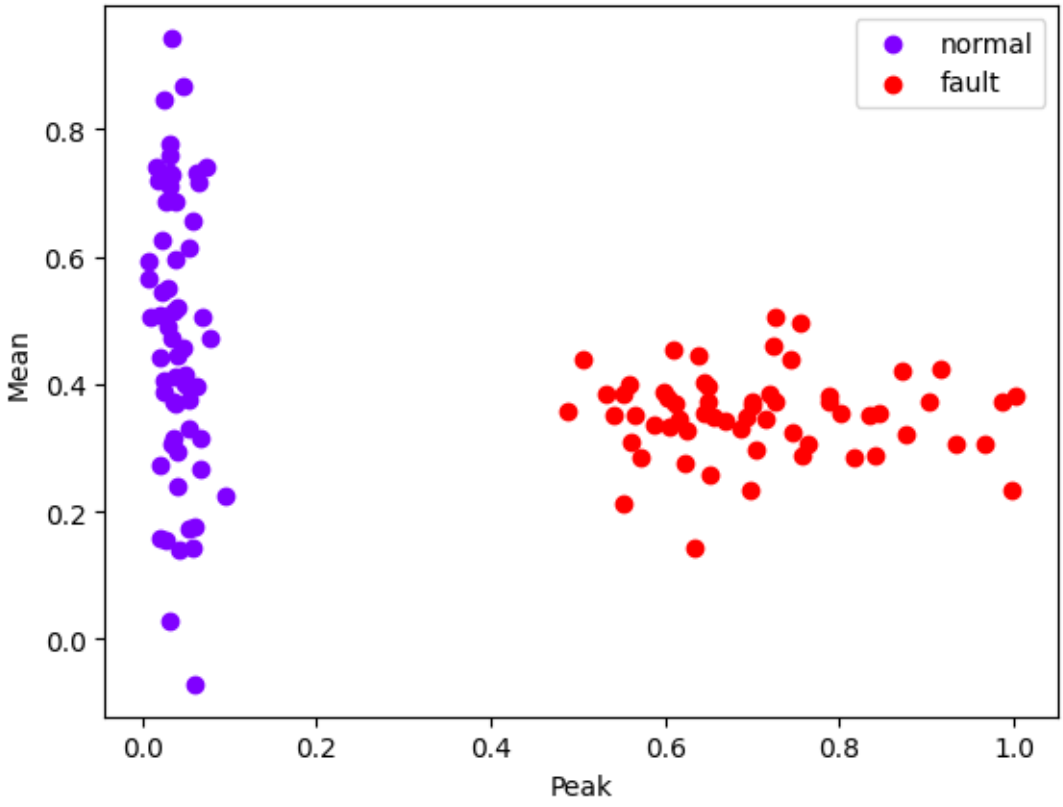

precision
1.00
0.99
0.98
0.97
0.96
0.95
0.94
precision
0
10
20
30
40
50
epoch
train
test


recall
train
test
1.04
1.02
1.00
0.98
0.96
recall
0
10
20
30
40
50
epoch


Confusion Matrix of train Set
Normal
Fault
240
0
0
240
True label
Normal
Fault
Predicted label
200
150
100
50
0



همان‌طور که مشخص است نمودار تابع هزینه هم برای مجموعه‌داده آموزش و هم ارزیابی به‌صورت نزولی عمل کرده و تا نزدیکی مقدار صفر رفته است نکته حائز اهمیت در این نمودار، کم‌بودن مقدار تابع هزینه در همان epoch اول است که نشان می‌دهد این مجموعه‌داده از تفکیک‌پذیری خوبی برخوردار است.



نمودار فوق مربوط به ترسیم ویژگی میانگین به ویژگی حداکثر مقدار داخل پنجره برای مجموعه‌داده ارزیابی است و همان‌طور که مشخص است این مجموعه‌داده به‌صورت ذاتی از تفکیک‌پذیری مناسبی برخوردار است و نتایج به‌دست‌آمده از آموزش کاملاً درست هستند. تنها ابهام موجود مربوط به نمودار recall است که از ابتدا ۱ می‌باشد. دلیل رخ دادن این مسئله این است که مقدار این metric در انتهای epoch اول و بعد از بروزرسانی وزن‌های مدل محاسبه شده است بنابراین چون تفکیک پذیری مجموعه داده بالا است و مدل به راحتی در epoch اول به نتیجه مطلوب رسیده این معیار از همان ابتدا ۱ است.

برای اظهار نظر درمورد عملکرد مدل به روی مجموعه‌داده ارزیابی با استفاده از نمودار تابع هزینه مجموعه‌داده آموزش، می‌توان گفت که با کاهش سریع تابع هزینه در ابتدا، بهبود عملکرد مدل روی مجموعه‌داده ارزیابی که مبدائی یکسان با مجموعه‌داده آموزش دارد، انتظار می‌رود؛ اما به‌صورت دقیق نمی‌توان این مسئله را مشخص کرد و حتی ممکن است در شرایط خاص برای مجموعه‌داده ارزیابی نتیجه‌گیری اشتباه باشد. برای رفع این مشکل می‌توان یک قسمت از مجموعه‌داده آموزش را به‌عنوان مجموعه‌داده صحت‌سنجی انتخاب کرد و با بررسی آن روی عملکرد مدل روی مجموعه‌داده ارزیابی نظر داد.

۴. فرآیند آموزش و ارزیابی را با استفاده از یک طبقه بند خطی آمادهٔ پایتون (در model_linear.sklearn) انجام داده و نتایج را مقایسه کنید. در حالت استفاده از دستورات آمادهٔ سایکیت لرن، آیا راهی برای نمایش نمودار تابع اتلاف وجود دارد؟ پیاده سازی کنید

```python
from sklearn.linear_model import SGDClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

perceptron = SGDClassifier(loss='perceptron', learning_rate='adaptive', eta0=0.01, random_state=53)

# Train the model
perceptron.fit(X_train, y_train)

# Make predictions
y_pred = perceptron.predict(X_test)

# Calculate and print the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')

```

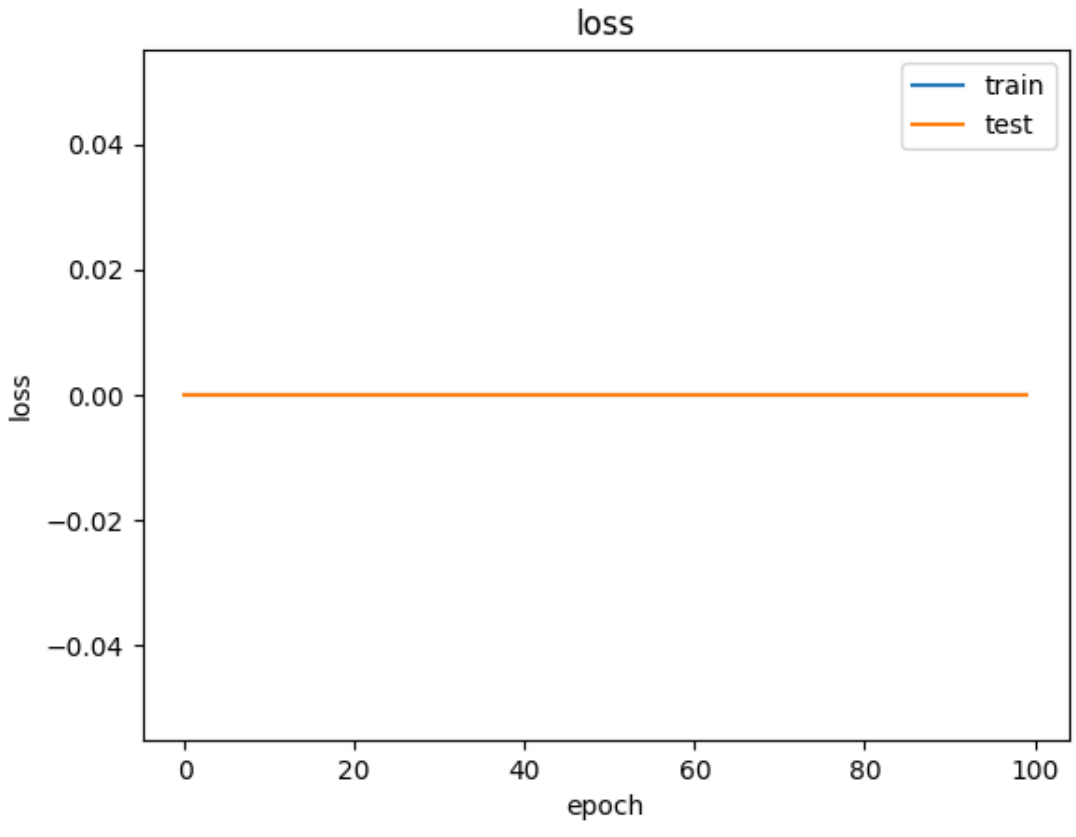
```
Accuracy: 1.0
```

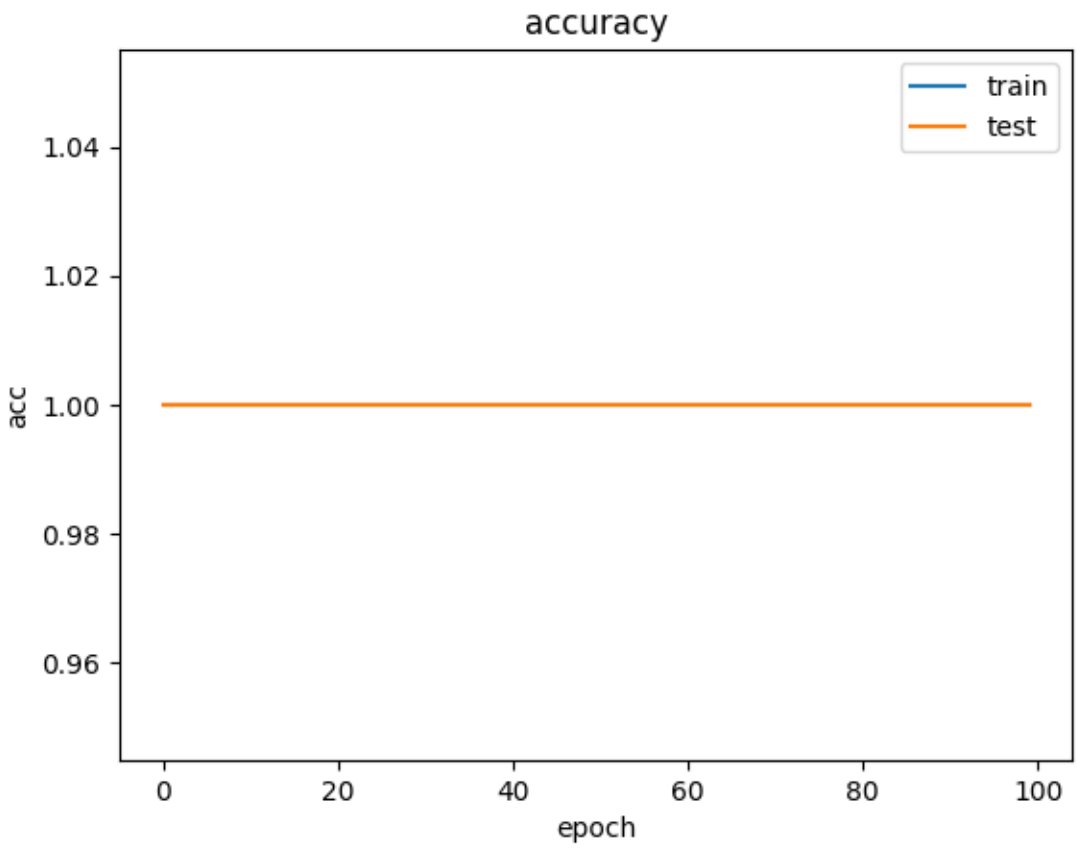
با استفاده از کد فوق و کتابخانه sklearn، مدل perceptron آموزش دیده است که همانند نتیایج قبلی به دقت ۱ رسیده است. برای ثبت نتایج هر epoch نیاز است تا از متد partial_fit استفاده کنیم. این متد با هربار فراخوانی یک epoch مدل را آموزش می‌دهد و می‌توان بعد از آموزش مقادیر هزینه را ثبت نمود.

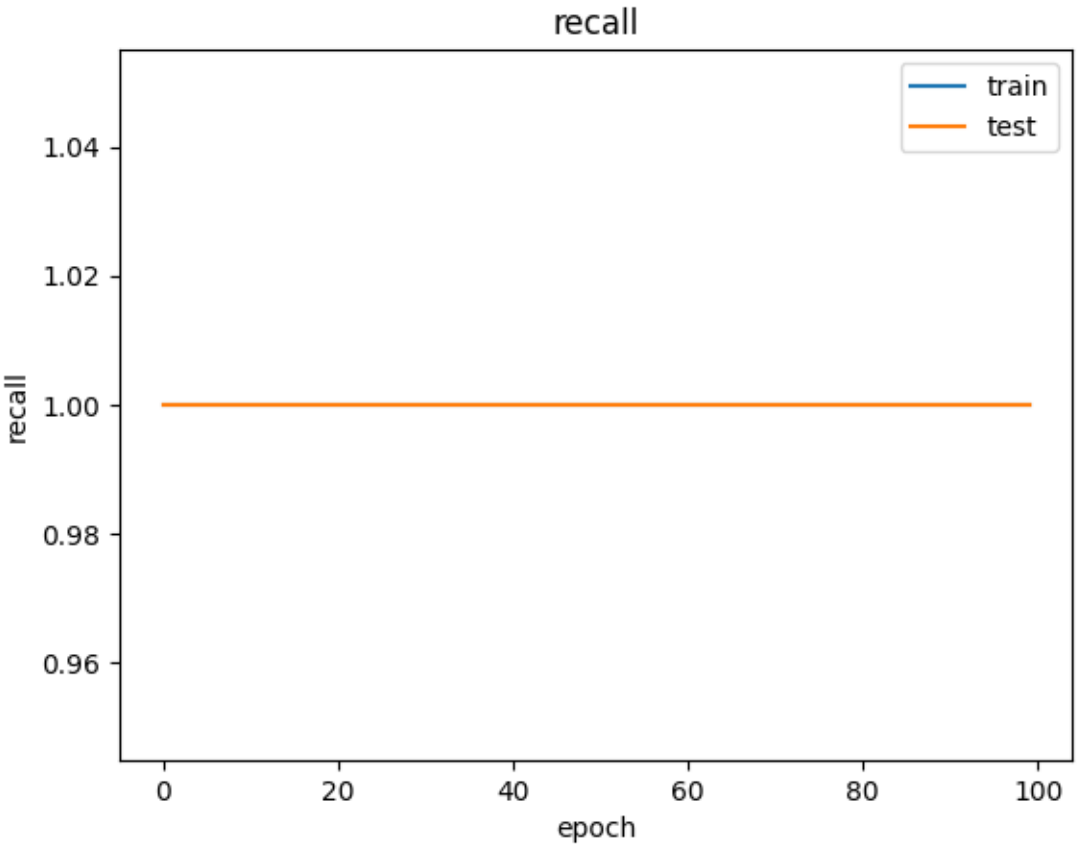
```python
# Training loop
for epoch in range(n_epochs):
    # Update the model with the current training data
    model.partial_fit(X_train, y_train, classes=np.unique(y_train))

    # Make predictions
    y_pred_train = model.predict(X_train)

    # Calculate accuracy, recall, and precision
    accuracy = accuracy_score(y_train, y_pred_train)
    recall = recall_score(y_train, y_pred_train, average='macro')
    precision = precision_score(y_train, y_pred_train, average='macro')

    # Calculate hinge loss
    # The decision function gives the distance of the samples to the hyperplane
    decision_scores = perceptron.decision_function(X_train)
    loss = ((y_pred_prob_train-np.array(y_train))**2).mean()

    # Save the metrics
    acc_list.append(accuracy)
    recall_list.append(recall)
    precision_list.append(precision)
    loss_list.append(loss)
```

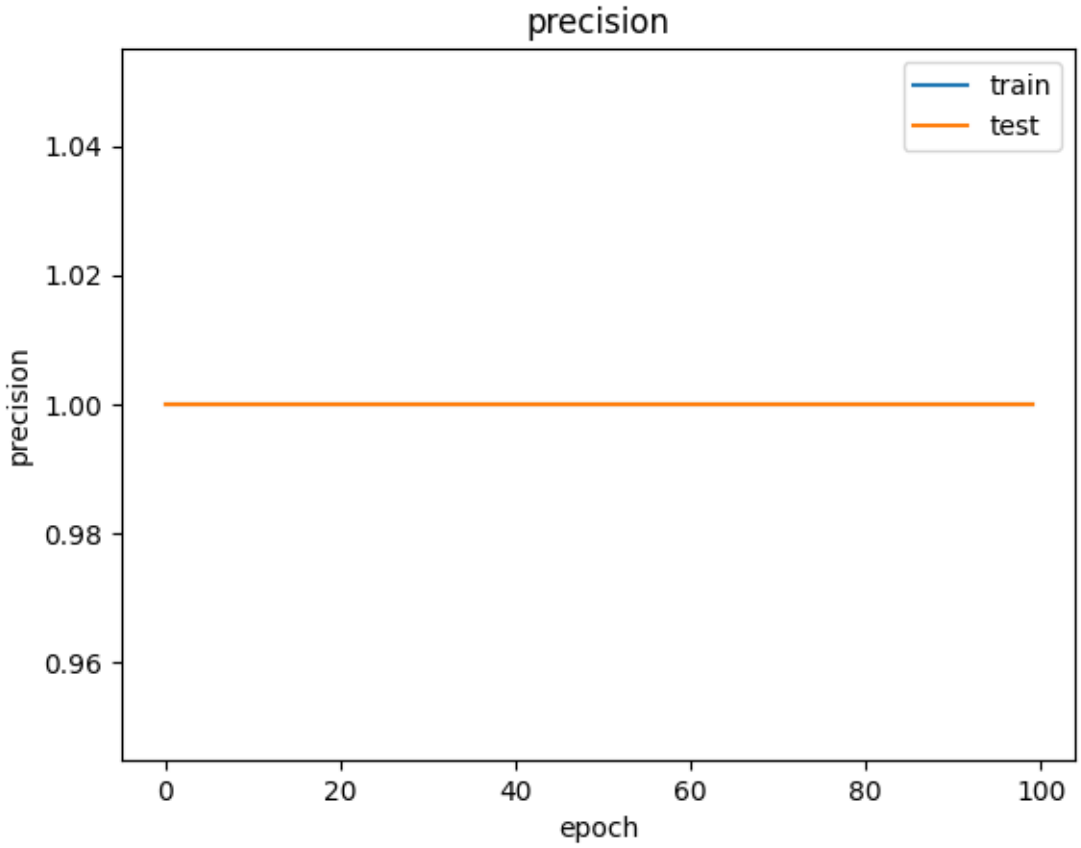
کد فوق نمایش روش استفاده از partial_fit را نشان می‌دهد که در ادامه نتایج ذخیره شده در این روش، نشان داده شده است.

در ادامه نمودارهای مربوط به این آموزش نمایش داده شده است.




accuracy
train
test
1.04
1.02
1.00
0.98
0.96
acc
0
20
40
60
80
100
epoch


recall
train
test
1.04
1.02
1.00
0.98
0.96
recall
0
20
40
60
80
100
epoch

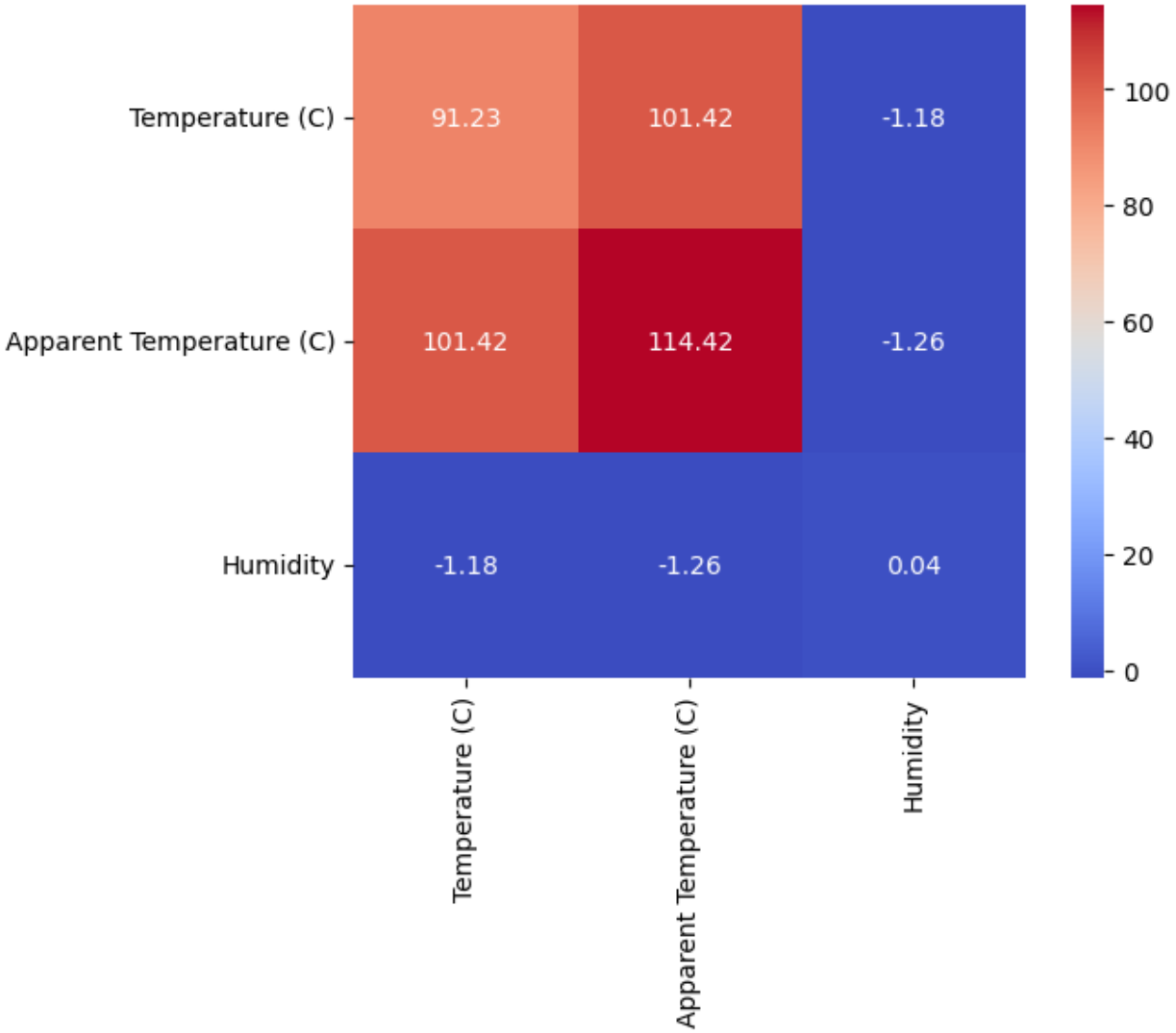


همانطور که مشخص است از اولین epoch نتیجه تابع هزینه این آموزش هم برای مجموعه‌داده آموزش و هم برای ارزیابی به به صفر رسیده و تمام معیارها به مقدار ۱ رسیده‌اند علت تفاوت این نمودار با آموزش قبلی این است که الگوریتم‌های کتابخانه sklearn بهینه‌هستند و در همان epoch اول به بهترین میزان خود رسیده است.

سوال ۳

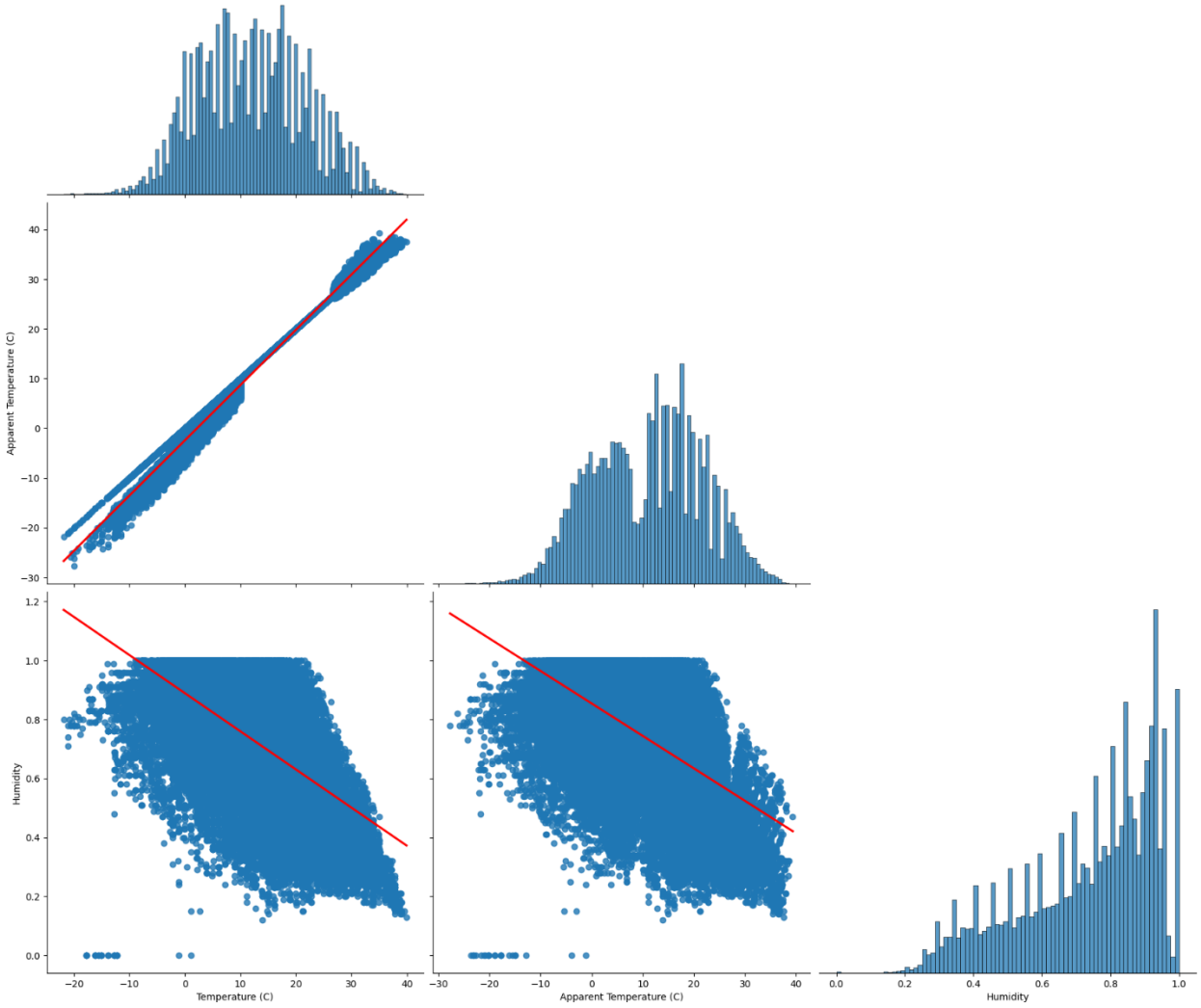
یک دیتاست در زمینهٔ آب و هوا با نام Szeged in Weather ۲۰۰۶-۲۰۱۶ را در نظر بگیرید. در این دیتاست هدف آن است که ارتباط بین Humidity با Temperature و هم چنین ارتباط بین Humidity و Temperature Apparentپیدا شده و با کمک داده های Humidity و Temperature تخمین انجام شود.

۱. ابتدا هیت مپ ماتریس همبستگی و هیستوگرام پراکندگی ویژگی ها را رسم و تحلیل کنید

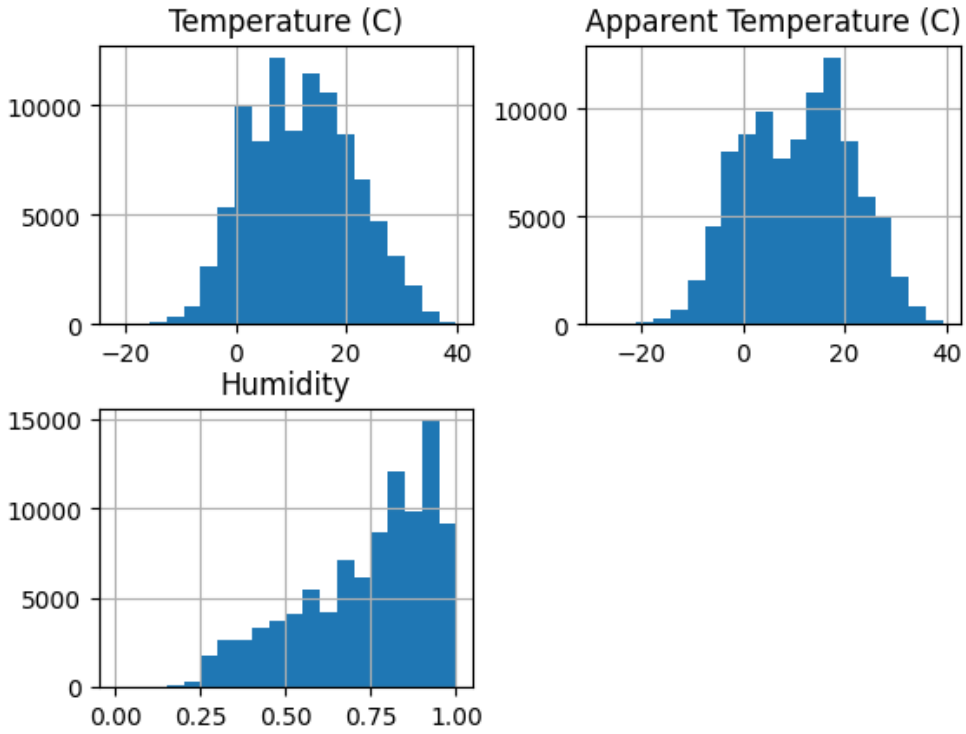


نمودار فوق مربوط به heat map داده‌های موجود است و همان‌طور که مشخص است واریانس رطوبت بسیار کم است که این به معنی تغییرات کم رطوبت هوا در زمان‌های متفاوت در محل اندازه‌گیری است. از سوی دیگر واریانس دما و دمای ظاهری در طول ساعات متفاوت بسیار زیاد است که این مسئله باتوجه‌به تفاوت دما در طول روز و شب به علت حضور خورشید یا شرایط متفاوت جوی مانند وزیدن باد منطقی

است. از سوی دیگر با توجه‌به منفی بودن ضریب covariance مشخص است که افزایش دمای هوا و دمای قابل احساس، تاثیر منفی روی رطوبت هوا دارد و عکس این رابطه نیز صحیح می‌باشد. ضریب covariance مربوط به دما و دمای حسی مقداری مثبت و بزرگ می‌باشد که نشان دهنده این است که افزایش هریک از این متغییرها تاثیر مستقیم و زیادی روی دیگری خواهد گذاشت.



نمودار فوق مربوط به رسم نقاط دو مولفه از میان دما، دمای قابل احساس و رطوبت نسبت به همدیگر می‌باشد و نمودارهای روی قطر هیستوگرام پراکندگی داده می‌باشد باید توجه کرد که محورها نرمال شده‌اند.



نمودار فوق هیستوگرام حسگرها به صورت نرمال نشده است و همانطور که مشخص است چون رطوبت مقداری بین ۰ و ۱ دارد میزان واریانس آن به نسبت کم شده است. از طرف دیگر نمودارهای دما نشان‌دهنده توضیحی نرمال است.

۲. روی این دیتاست، تخمین LS و RLS را با تنظیم پارامترهای مناسب اعمال کنید. نتایج به دست آمده را با محاسبهٔ خطاها و رسم نمودارهای مناسب برای هر دو مدل با هم مقایسه و تحلیل کنید.

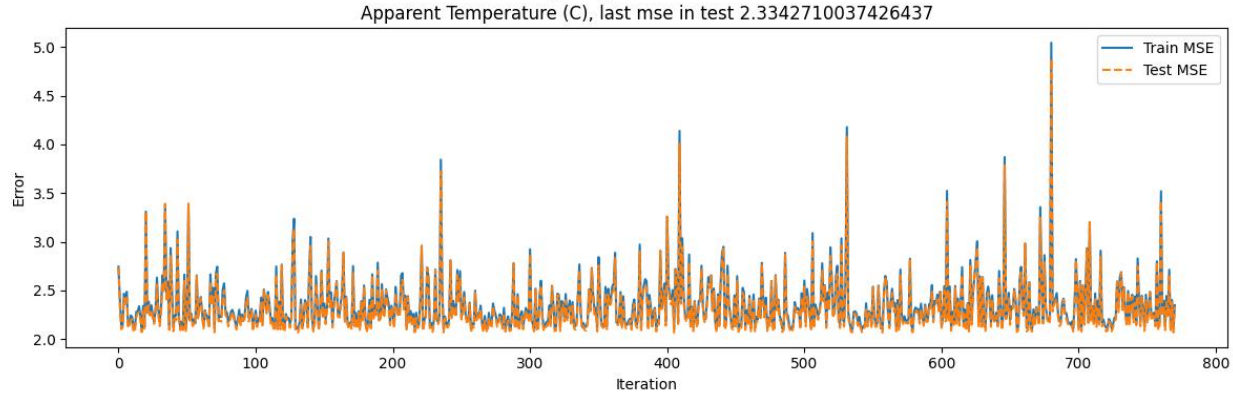
با استفاده از کتابخانه sklearn مدل LinearRegression را پیاده سازی شده است و مجموعه‌داده با نسبت ۰.۲ برای مجموعه داده ارزیابی و ۰.۸ برای مجموعه آموزش جدا شده است و نتایج آن به شرح زیر است.

```
Apparent Temperature (C) accuracy on training set is equal to 0.9863194080189283
Apparent Temperature (C) accuracy on testset is equal to 0.9864954840061168
Apparent Temperature (C) mse loss on training set is 1.5705913008265022
Apparent Temperature (C) mse loss on test set is 1.5244808554474927

Humidity accuracy on training set is equal to 0.4424797955038111
Humidity accuracy on testset is equal to 0.44192120432173676
Humidity mse loss on training set is 0.021310949624329792
Humidity mse loss on test set is 0.021289415712958348

Temperature (C) accuracy on training set is equal to 0.9871057929817124
Temperature (C) accuracy on testset is equal to 0.9872771524102542
Temperature (C) mse loss on training set is 1.1815381381847463
Temperature (C) mse loss on test set is 1.1402114757257233
```

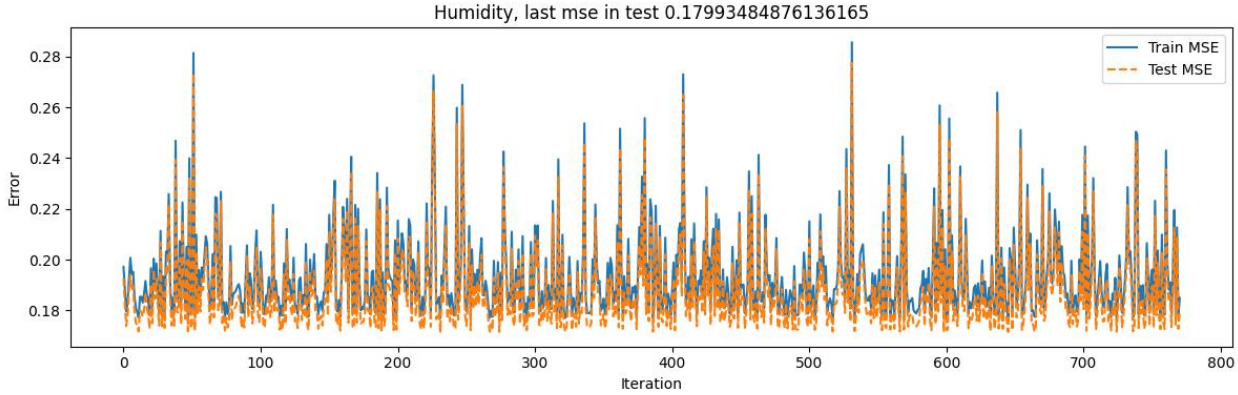
در ادامه الگوریتم RLS پیاده سازی شده و بهترین میزان برای forgetting factor ابتدا از بین ۰.۱ تا ۰.۹ با گام ۰.۱ بدست آمد و سپس از بین ۰.۹ تا ۰.۹۹ با گام ۰.۰۱ انتخاب شد که بهترین نتیجه به ازای ۰.۹۹ می‌باشد.

forgetting factor=0.9

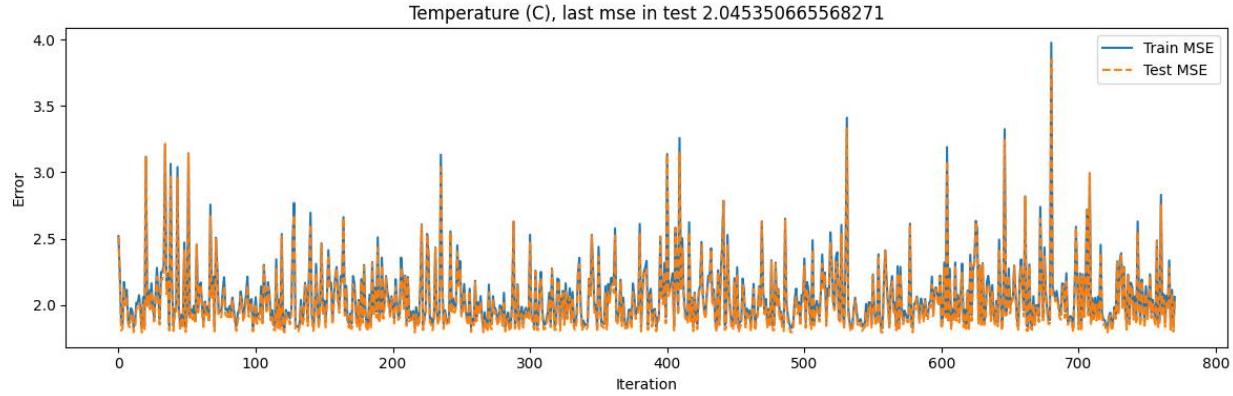
Apparent Temperature (C), last mse in test 2.3342710037426437

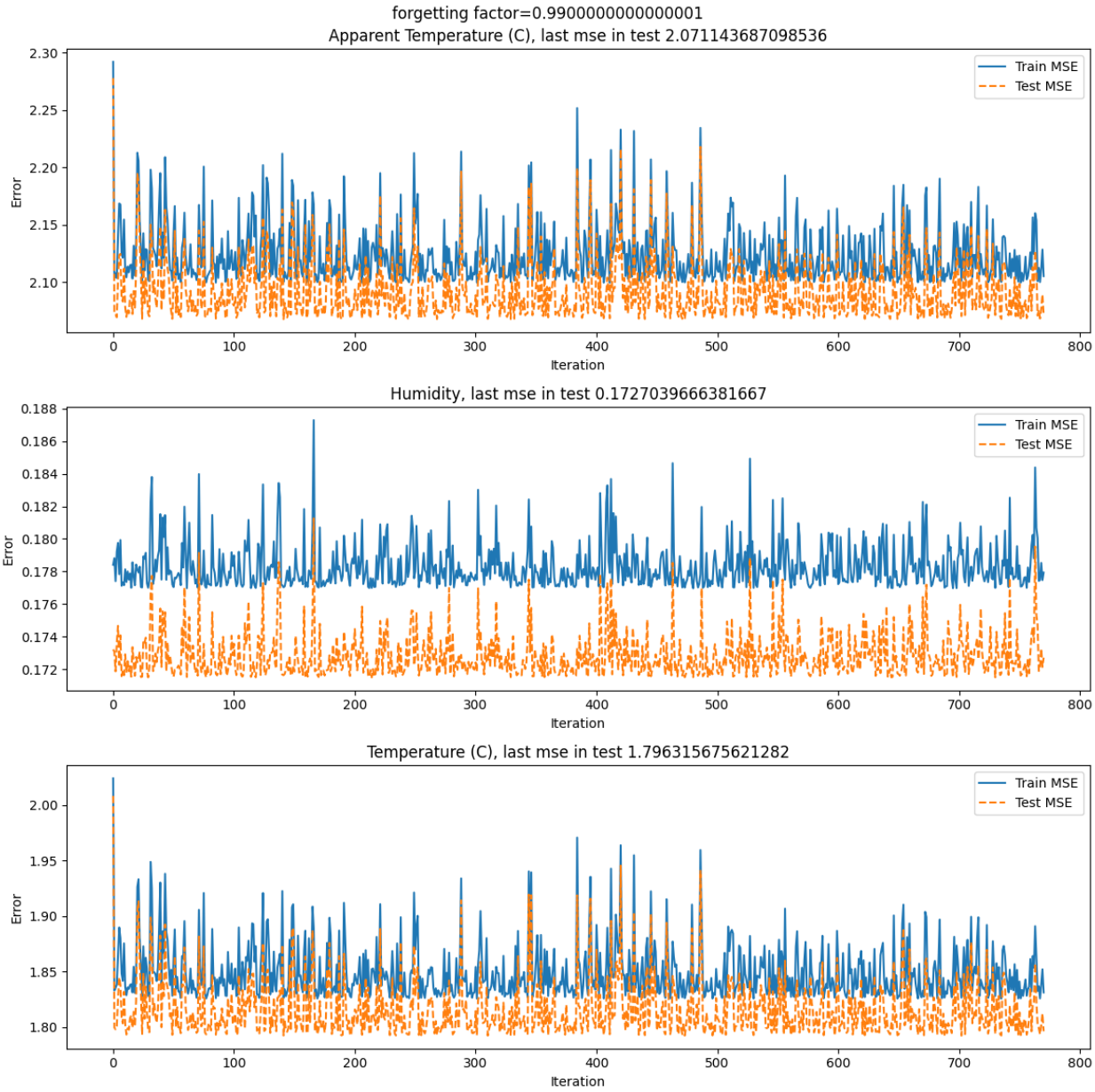


Humidity, last mse in test 0.17993484876136165



Temperature (C), last mse in test 2.045350665568271





نمودارهای فوق مربوط به forgetting factorهای ۰.۹ و ۰.۹۹ است و بقیه نمودارها در فایل Q3.ipynb می‌باشد. همانطور که مشخص است میزان mse برای LS کمتر از RLS می‌باشد بنابراین در این مقایسه الگوریتم LS عملکرد بهتری دارد.

۳. در مورد Square Least Weighted توضیح دهید و آن را روی دیتاست داده شده ا عمال کنید.

Weighted least square بر پایه least square بنا شده و با اضافه کردن یک ماتریس مربعی به هر نمونه از داده‌های جمع‌آوری شده وزنی مشخص را نسبت می‌دهد که این مسئله می‌تواند بعضی ایرادات کمی مجموعه داده را برطرف کنند به عنوان نمونه در یک مجموعه داده نامتعادل با افزایش وزن داده‌هایی که کمتر هستند یا افزایش وزن داده‌هایی که از اهمیت بیشتری برخوردار هستند می‌توان به نتیجه مطلوب‌تری رسید. رابطه مربوط به WLS به شکل زیر است.

$$\theta = (X^T W X)^{-1} X^T W y$$

در حل این سوال به منظور لحاظ کردن عنصر زمان در محاسبه بهترین وزن‌ها، به هر نمونه از مجموعه داده وزنی معکوس شماره مکانی آن داده شده است. نتایج آموزش به شرح زیر می‌باشد.

```
mse loss for Apparent Temperature (C) 1.5319736086224753
mse loss for Humidity 0.022276381331441112
mse loss for Temperature (C) 1.1464081578485372
```

همانطور که مشخص است این مدل عملکرد بهتری نسبت به RLS دارد اما با اختلاف کمی از LS عملکرد ضعیف‌تری دارد.

[i] https://engineering.case.edu/bearingdatacenter

[ii] XiongMeijing/CWRU-1: Multiclass classification of vibration signals of faulty bearings (github.com)

[iii] Yoo, Y., Jo, H. and Ban, S.W., 2023. Lite and efficient deep learning model for bearing fault diagnosis using the CWRU dataset. *Sensors*, *23*(6), p.3157.