دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکدهٔ مهندسی برق - گروه مهندسی کنترل

درس یادگیری ماشین

گزارش مینی پروژه ۱

نام و نام خانوادگی	دانیال عبداللهی نژاد
شمارهٔ دانشجویی	۴۰۱۰۹۱۶۴
تاریخ	فروردین‌ماه ۱۴۰۳

فهرست مطالب

فهرست تصاویر

۱ سوال اول

۱.۱

برای تبدیل یک فرآیند طبقه‌بندی دو کلاسه به چند کلاسه، باید تغییراتی ایجاد کنیم تا از پیچیدگی و نیازهای طبقه‌بندی چند کلاسه پشتیبانی کنیم.

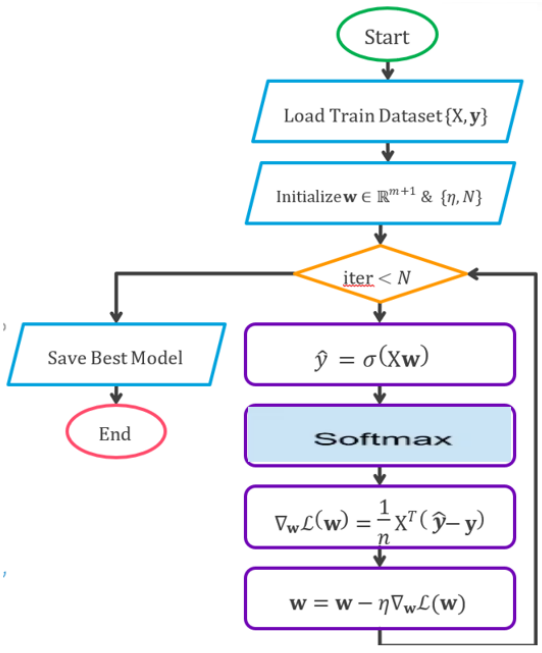
در شروع، مجموعه داده آموزشی X, y به روال معمول بارگذاری می‌شود، با این تفاوت که 'y' حالا می‌تواند شامل برچسب‌های متعدد کلاس به جای تنها دو حالت باشد. در مرحله اولیه، بجای ایجاد یک بردار وزن برای نتایج دوگانه، یک ماتریس وزن در فضای $\mathbb{R}^{(m+1)\times K}$ ساخته می‌شود، که در آن 'm' تعداد ویژگی‌ها و 'K' تعداد کلاس‌ها را نشان می‌دهد.

تغییر مهم بعدی در مرحله پیش‌بینی رخ می‌دهد. در حالی که برای طبقه‌بندی دو کلاسه معمولاً از تابع سیگموئید σ استفاده می‌شود، در سناریوی چند کلاسه، تابع softmax جایگزین آن می‌شود.

پس از پیش‌بینی، تابع خطا نیز برای حالت چند کلاسه محاسبه مجدد می‌شود. در حالی که طبقه‌بندی دو کلاسه از binary cross entropy استفاده می‌کند، طبقه‌بندی چند کلاسه categorical cross-entropy است که با توزیع واقعی و پیش‌بینی شده مقایسه می‌کند.

گرادیان خطا و به‌روزرسانی وزن‌ها مشابه روش طبقه‌بندی دو کلاسه محاسبه می‌شود. این فرآیند تکراری ادامه پیدا می‌کند تا زمانی که حداکثر تعداد تکرارها به پایان برسد یا به معیار دیگری برای توقف برسد. در پایان آموزش، بهترین مدل که خطای دسته‌ای را کمینه می‌کند، ذخیره می‌شود.

به طور خلاصه، در حالی که ساختار کلی کار طبقه‌بندی مشابه باقی می‌ماند، تفاوت‌های اساسی در کار با طبقه‌بندی چند کلاسه در مراحل پیش‌بینی، تابع خطا، و ابعاد ماتریس وزن هستند. این تغییرات به مدل امکان می‌دهند تا بین بیش از دو کلاس تمایز قائل شود و پیش‌بینی‌های دقیقی برای هر کلاس در یک مجموعه داده انجام دهد.



شکل ۱: دیاگرام آموزش چندکلاسه

۲.۱

تابع make_classification در scikit learn برای تولید داده یرای طبقه‌بندی استفاده می‌شود. این تابع یک مجموعه داده را ایجاد می‌کند که می‌توان برای آموزش و آزمایش الگوریتم‌های یادگیری ماشین استفاده کرد. در زیر پارامترهای کلیدی و اهداف آن‌ها آورده شده است:

- n_samples: تعداد نمونه‌هایی که باید تولید شود، به طور اساسی اندازه مجموعه داده است.
- n_features: تعداد کل ویژگی‌ها، که شامل ویژگی‌های غنی (informative) ، اضافی و تکراری است.
- n_informative: تعداد ویژگی‌های غنی که برای ساخت مدل مفید هستند.
- n_redundant: تعداد ویژگی‌های اضافی، که به عنوان ترکیب‌های خطی از ویژگی‌های اطلاعاتی تولید می‌شوند.
- n_repeated: تعداد ویژگی‌هایی که تکرار ویژگی‌های غنی و اضافی هستند.
- n_classes: تعداد کلاس‌ها یا برچسب‌ها در مسئله طبقه‌بندی.
- n_clusters_per_class: تعداد خوشه‌ها در هر کلاس، که بر توزیع داده‌ها در هر کلاس تأثیر می‌گذارد.
- weights: نسبت‌های نمونه‌ها برای هر کلاس. در واقع به نوعی وزن هر کلاس را مشخص می‌کند.
- flip_y: نسبت نمونه‌هایی که برچسب کلاس‌های آن‌ها به طور تصادفی معکوس می‌شود، از آن برای وارد کردن نویز به داده می‌توان استفاده کرد.
- class_sep: ضریبی برای فاصله بین خوشه‌های کلاس، هرچه ضریب بزرگتر باشد میزان جدا بودن داده‌ها بیشتر است.
- hypercube: اگر true باشد، خوشه‌ها در رأس‌های یک چندوجهی قرار می‌گیرند؛ اگر false باشد، به طور تصادفی قرار می‌گیرند.
- shift: داده‌ها را با مقدار ثابتی شیفت می‌دهد.
- scale: داده‌ها را در یک ضریب مشخص ضرب می‌کند که این امر می‌تواند برای محدود کردن فضای داده‌ها استفاده شود.
- shuffle: برای ترکیب داده‌ها پس از تولید آن‌ها استفاده می‌شود.

با توجه به پارامترهای موجود در این تابع و صورت سوال، یک سری داده به تعداد ۱۰۰۰ نمونه و ۴ کلاس که شامل ۳ ویژگی است تولید می‌کنیم. می‌توان با تغییر پارامترهای ذکر شده داده‌ها را چالش برانگیز کرد. به طور مثال اگر هر ۳ ویژگی از نوع informative باشند، ویژگی‌ها به یکدیگر وابستگی ندارند که می‌تواند باعث ایجاد چالش در داده شود. پارامتر دیگری که می‌تواند در این امر اثرگذار باشد پارامتر وزن است که باعث ایجاد ناتعادلی بین کلاس‌ها شود و این موضوع دادها چالشی‌تر می‌کند. پارامترهای flip_y و class_sep که به ترتیب باعث ایجاد نویز و میزان جدا بودن کلاس‌ها می‌شوند نیز تاثیر بسزایی در میزان پیچیدگی داده ما خواهند داشت. همچنین سه پارامتر shift, scale, shuffle با توجه به توضیحات فوق در چالشی کردن داده اثر گذارند.

دیتای تولید شده بر اساس کد زیر است:

مشاهده می‌شود که تلاش شده از پارامترهای مختلف برای چالش برانگیز کردن دیتا استفاده شود. نمایش داده‌ها بر اساس ترکیب ویژگی‌های مختلف برای نمایش در دو بعد و همچنین نمایش سه بعدی آن به صورت زیر است:

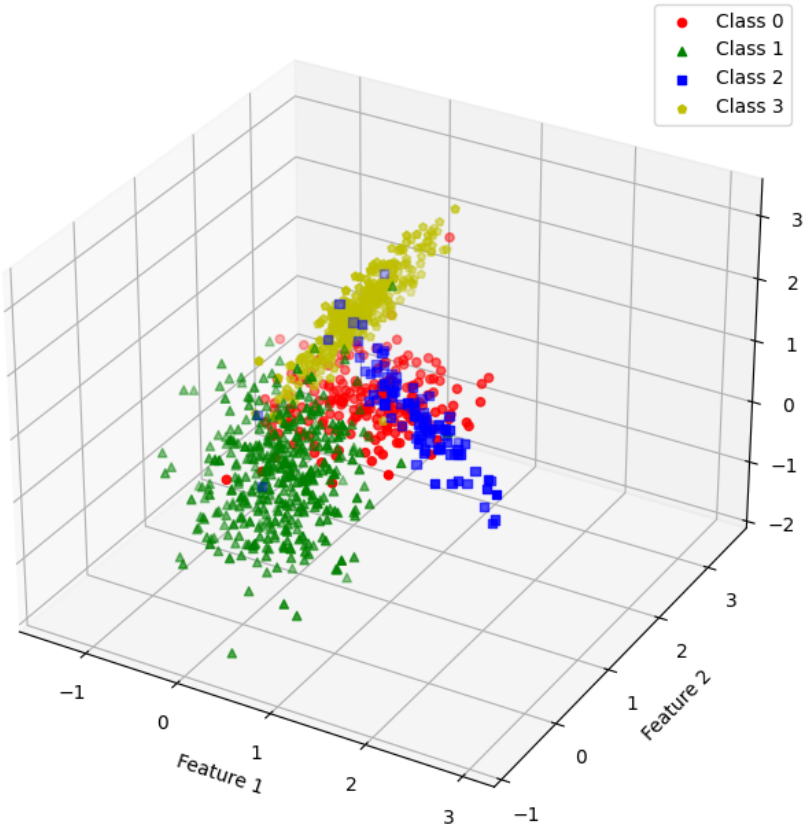
```python
X, y = make_classification(n_samples=1000,
                           n_features=3,
                           n_informative=3,
                           n_redundant=0,
                           n_repeated=0,
                           n_classes=4,
                           n_clusters_per_class=1,
                           weights=[0.2, 0.4, 0.1, 0.3],
                           flip_y=0.01,
                           class_sep=1.5,
                           hypercube= True,
                           shift=2,
                           scale=0.5,
                           shuffle=True,
                           random_state=64)
```

شکل ۲: کد تولید دیتا

شکل ۳: نمایش دو بعدی داده‌ها بر اساس جفت ویژگی‌های مختلف

۳.۱

با بررسی مدل‌های موجود در sklearn.linear_model از LogisticRegression و SGDClassifier برای طبقه‌بندی داده‌ها استفاده می‌کنیم. در ابتدا با استفاده از train_test_split داده‌ها را به دو دسته آموزش و ارزیابی تقسیم می‌کنیم. برای این منظور از نسبت ۷۵ به ۲۵ استفاده می‌کنیم. سپس با استفاده از مدل‌های طبقه‌بندی ذکر شده فرایند یادگیری را با داده آموزش انجام می‌دهیم و سپس ارزیابی را با استفاده از داده تست بررسی می‌کنیم.



شکل ۴: نمایش سه بعدی داده تولید شده

```
from sklearn.linear_model import LogisticRegression, SGDClassifier

model1 = LogisticRegression(solver='sag', max_iter=200, random_state=64)
model1.fit(X_train, y_train)

model2 = SGDClassifier(loss='log_loss', random_state=64)
model2.fit(X_train, y_train)
```

```
SGDClassifier
SGDClassifier(loss='log_loss', random_state=64)
```

شکل ۵: کد مربوط به یادگیری مدل‌ها

حال با استفاده از معیارهای ارزیابی متفاوت دقت مدل‌ها را بررسی می‌کنیم.

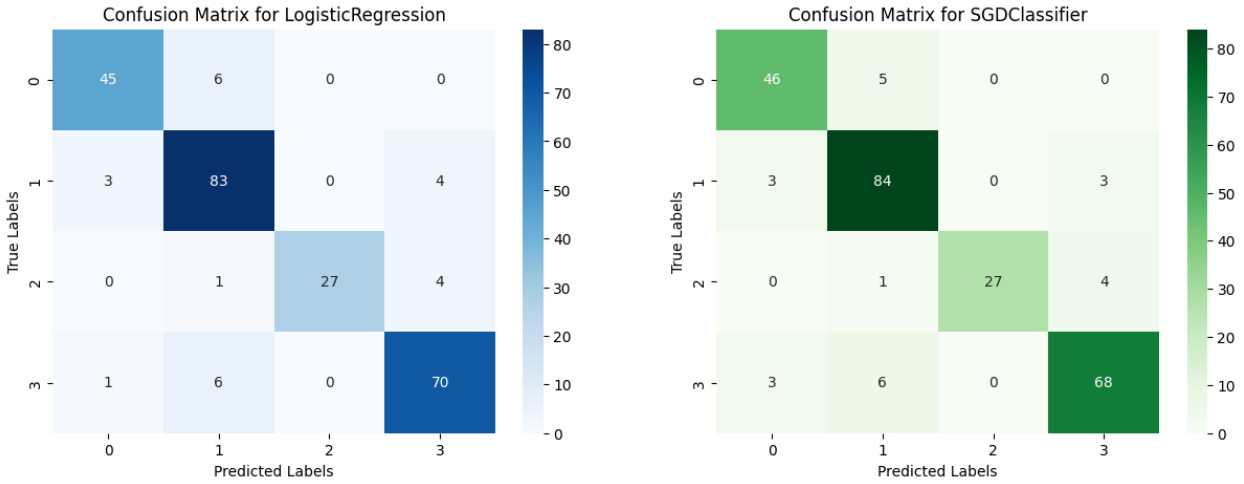
```
[107] model1.score(X_train, y_train)
0.94

[108] model1.score(X_test, y_test)
0.9

[109] model2.score(X_train, y_train)
0.9213333333333333

model2.score(X_test, y_test)
0.9
```

شکل ۶: دقت مدل‌ها

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
model1_report = classification_report(y_test,yhat1)
model2_report = classification_report(y_test,yhat2)
print(model1_report)
print(model2_report)
```

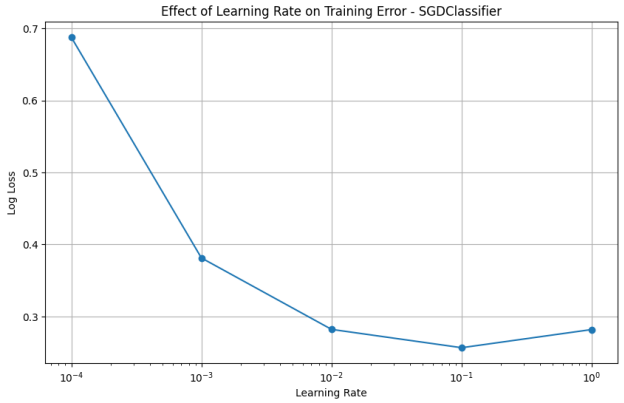
```
              precision    recall  f1-score   support

           0       0.92      0.88      0.90        51
           1       0.86      0.92      0.89        90
           2       1.00      0.84      0.92        32
           3       0.90      0.91      0.90        77

    accuracy                           0.90       250
   macro avg       0.92      0.89      0.90       250
weighted avg       0.90      0.90      0.90       250

              precision    recall  f1-score   support

           0       0.88      0.90      0.89        51
           1       0.88      0.93      0.90        90
           2       1.00      0.84      0.92        32
           3       0.91      0.88      0.89        77

    accuracy                           0.90       250
   macro avg       0.92      0.89      0.90       250
weighted avg       0.90      0.90      0.90       250
```

شکل ۷: معیارهای متفاوت ارزیابی (LogisticRegression بالا و SGDClassifier پایین قرار دارد)

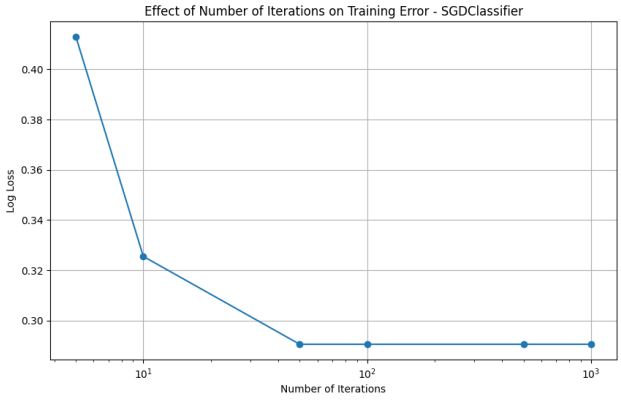


شکل ۸: Confusion Matrices

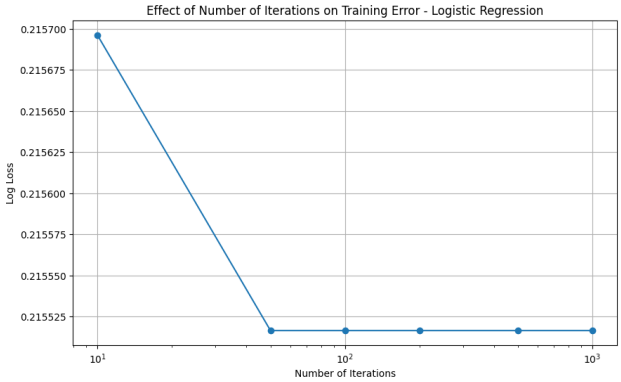
مشاهده می‌شود که LogisticRegression مقداری بهتر از SGDClassifier عمل می‌کند. حال برای بهبود مدل از تکنیک‌های مختلفی بهره می‌بریم. در ابتدا یک فرایند پیش پردازش روی داده انجام می‌دهیم. در این قسمت از نرمال‌سازی داده برای بهبود عملکرد مدل استفاده می‌کنیم. روش نرمال‌سازی استانداردسازی در این قسمت انجام می‌شود. مورد بعدی که در بهبود مدل تاثیر دارد تعداد تکرار مدل و نرخ یلدگیری است. برای این منظور تاثیر این پارامترها را بر خطای آموزش بررسی می‌کنیم. باید توجه کرد که در مدل LogisticRegression با توجه به solver ها پارامتر نرخ آموزش قابل تعیین نیست.



شکل ۹: Effect of Learning Rate on Training Error - SGDClassifier



شکل ۱۰: Effect of Number of Iterations on Training Error - SGDClassifier



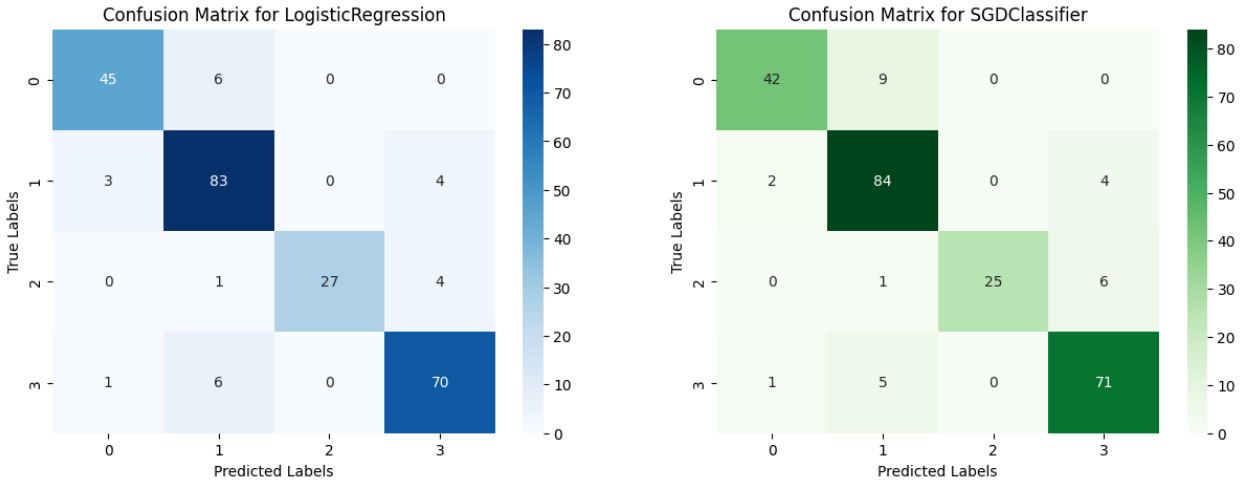
شکل ۱۱: Effect of Number of Iterations on Training Error - Logistic Regression

با توجه به نتایج، مقادیر مناسب برای نرخ یادگیری و تعداد تکرار مشخص می‌شود. یکی دیگر از موازد مهم در بهبود عملکرد مدل پیدا کردن بهترین پارارمترها با توجه به داده است. برای این کار می‌توان از GridSearchCV استفاده کرد. این دستور، مجموعه‌ای از پارامترها را دریافت می‌کند و با بررسی آنها روی دیتا بهترین ترکیب را برای مدل پیدا می‌کند. از این روش برای پیدا کردن پارامترهای مناسب می‌توان استفاده کرد.

	precision	recall	f1-score	support
0	0.92	0.88	0.90	51
1	0.86	0.92	0.89	90
2	1.00	0.84	0.92	32
3	0.90	0.91	0.90	77
accuracy			0.90	250
macro avg	0.92	0.89	0.90	250
weighted avg	0.90	0.90	0.90	250

	precision	recall	f1-score	support
0	0.92	0.90	0.91	51
1	0.87	0.92	0.90	90
2	1.00	0.84	0.92	32
3	0.90	0.91	0.90	77
accuracy			0.90	250
macro avg	0.92	0.89	0.91	250
weighted avg	0.91	0.90	0.90	250

شکل ۱۲: نتایج پس از بهبود مدل



شکل ۱۳: Confusion Matrix پس از بهبود مدل

با توجه به نتایج در داده ارزیابی مدل SGDClassifier بهبود ۱ درصدی داشتیم. به طور کلی با توجه به محدودیت تعداد داده و عدم پیچیدگی زیاد این روش‌ها تاثیر قابل ملاحظه‌ای در نتایج ندارند اما بررسی این روش‌ها برای بهبود مدل مفید خواهد بود.

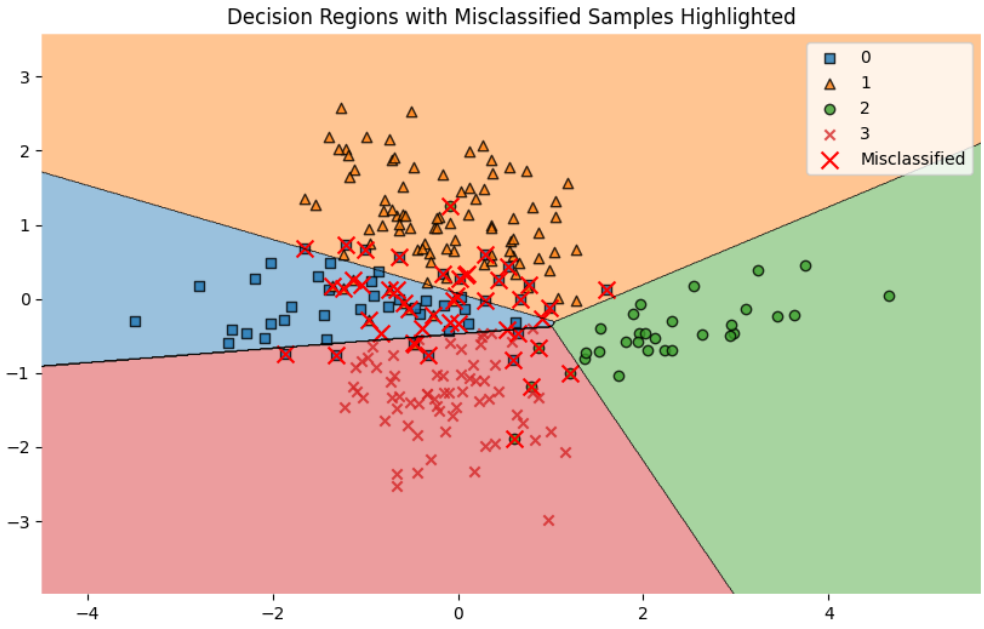
۴.۱

در این قسمت برای رسم نواحی تصمیم‌گیری از کتابخانه mlxtend استفاده می‌شود. با توجه به اینکه داده ما دارای ۳ ویژگی است، برای رسم نواحی تصمیم‌گیری در دو بعد نیاز به کاهش ابعاد داریم. برای این کار می‌توان از تکنیک PCA که به صورت آماده در sklearn

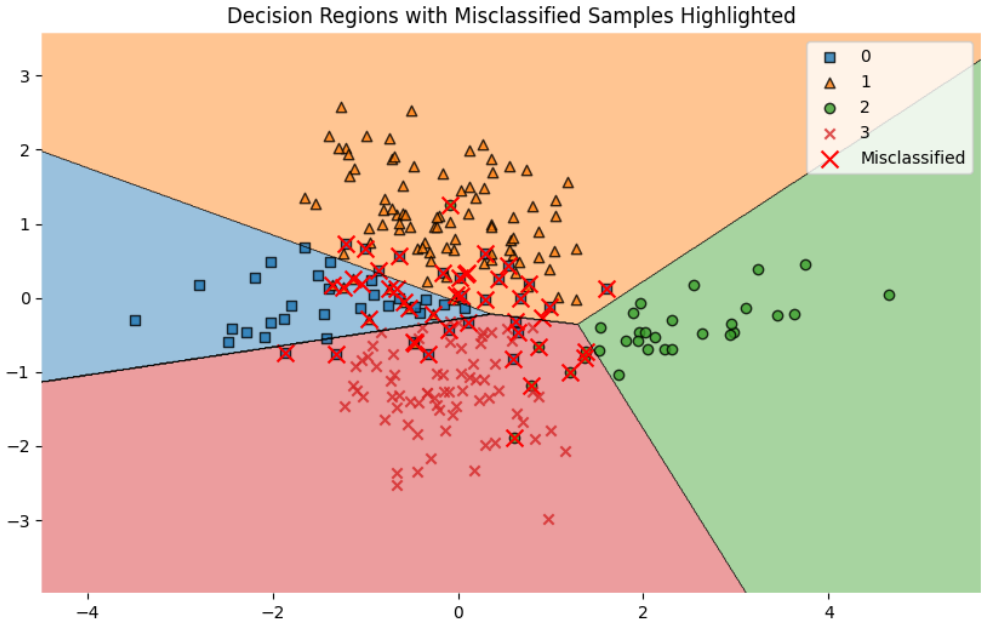
استفاده کرد. با کاهش ابعاد با استفاده از کد زیر نواحی تصمیم‌گیری را رسم می‌کنیم.

```
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
best_model1.fit(X_train_pca, y_train)

y_pred = best_model1.predict(X_test_pca)

misclassified = np.where(y_test != y_pred)[0]

plt.figure(figsize=(10, 6))
plot_decision_regions(X_test_pca, y_test, clf=best_model1, legend=2)

plt.scatter(X_test_pca[misclassified, 0], X_test_pca[misclassified, 1],
            color='red', label='Misclassified', marker='x', s=100)

plt.title('Decision Regions with Misclassified Samples Highlighted')
plt.legend()
plt.show()
```

شکل ۱۴: کد رسم نواحی تصمیم‌گیری



شکل ۱۵: نواحی تصمیم‌گیری مربوط به مدل LogisticRegression

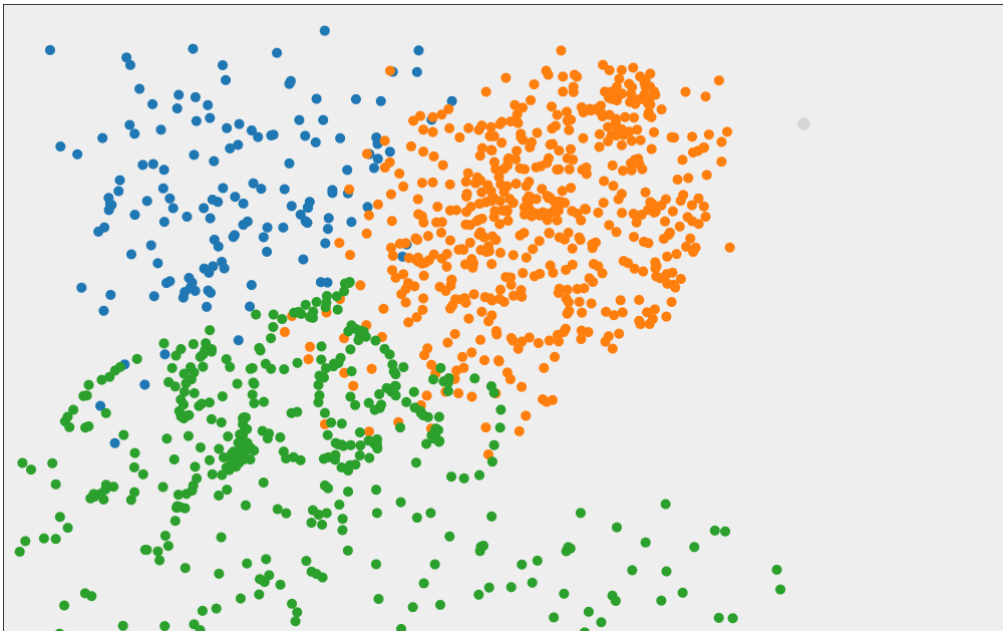


شکل ۱۶: نواحی تصمیم‌گیری مربوط به مدل SGDClassifier

در نمودارهای رسم شده نقاطی که اشتباه طبقه‌بندی شدند مشهودند و مشخص است که اشتباه در تصمیم‌گیری برای نقاطی که در مرز هستند عموما اتفاق می‌افتد. همچنین برخی نقاط به دلیل استفاده از پارامتر flip_y در تولید داده که به نوعی نویز هستند اشتباه طبقه‌بندی شده‌اند.

۵.۱

در این قسمت با استفاده از drawdata یک سری داده ایجاد می‌کنیم و مراحل قبل را روی داده جدید پیاده‌سازی می‌کنیم. داده تولید شده به صورت زیر است که شامل سه کلاس است و تلاش شده مقداری پیچیدگی آن را با ایجاد پراکندگی و تداخل زیاد کنیم.



شکل ۱۷: داده تولید شده با استفاده از drawdata

مشابه قسمت‌های قبل داده را به دو دسته آموزش و ارزیابی تقسیم می‌کنیم و فرایند یادگیری را با استفاده از مدل‌های LogisticRegression و SGDClassifier انجام می‌دهیم. نتایج بدست آمده به صورت زیر است.

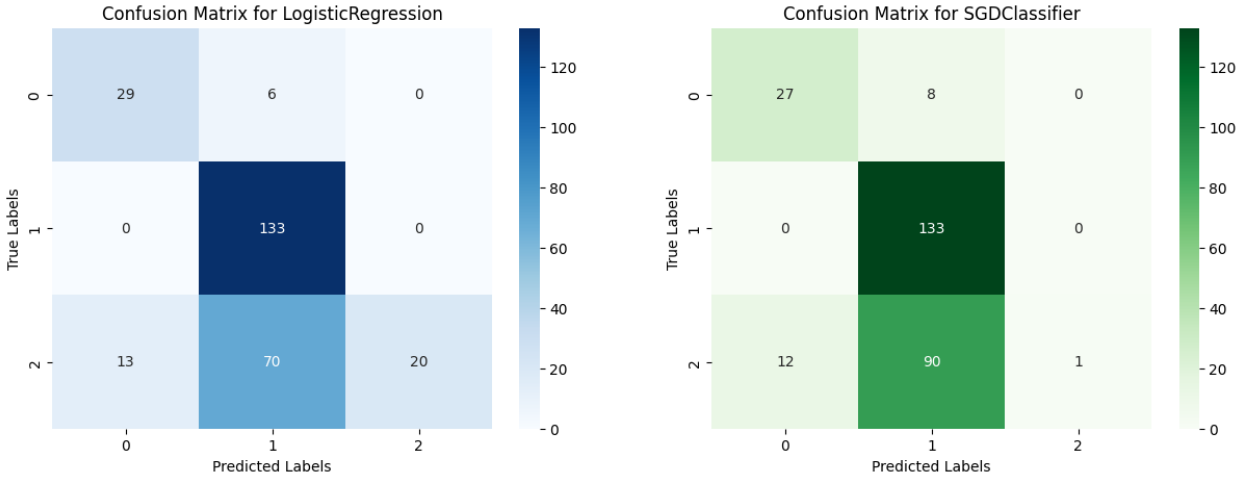
```
[22] model1_custom.score(X_train_custom, y_train_custom), model2_custom.score(X_train_custom, y_train_custom)
0s
     (0.6933497536945813, 0.6219211822660099)

[23] model1_custom.score(X_test_custom, y_test_custom), model2_custom.score(X_test_custom, y_test_custom)
     (0.6715867158671587, 0.5940959409594095)
```

شکل ۱۸: دقت مدل‌ها

```
model1_report_custom = classification_report(y_test_custom,yhat1_custom)
model2_report_custom = classification_report(y_test_custom,yhat2_custom)
print(model1_report_custom)
print(model2_report_custom)
```

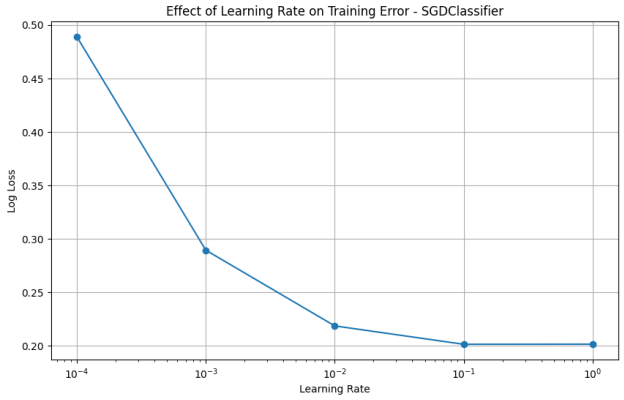
```
              precision    recall  f1-score   support

           a       0.69      0.83      0.75        35
           b       0.64      1.00      0.78       133
           c       1.00      0.19      0.33       103

    accuracy                           0.67       271
   macro avg       0.78      0.67      0.62       271
weighted avg       0.78      0.67      0.60       271

              precision    recall  f1-score   support

           a       0.69      0.77      0.73        35
           b       0.58      1.00      0.73       133
           c       1.00      0.01      0.02       103

    accuracy                           0.59       271
   macro avg       0.76      0.59      0.49       271
weighted avg       0.75      0.59      0.46       271
```

شکل ۱۹: معیارهای متفاوت ارزیابی (LogisticRegression بالا و SGDClassifier پایین قرار دارد)

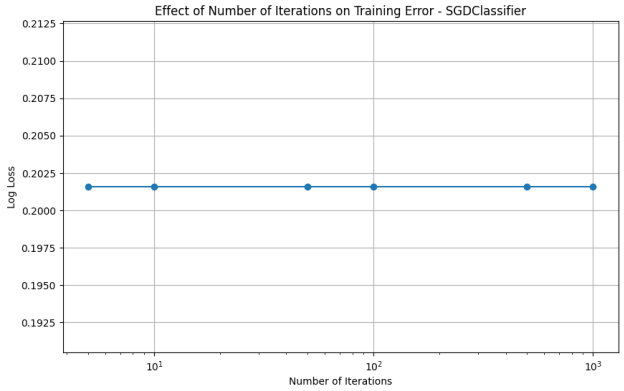


شکل ۲۰: Confusion Matrix

در این سری داده نیز مشخص است که LogisticRegression مقداری بهتر از SGDClassifier عمل می‌کند. با اعمال مواردی که در قسمت‌های قبل ذکر شد، تلاش می‌کنیم تا مدل را بهبود دهیم. بنابراین نتایج به شرح زیر است.

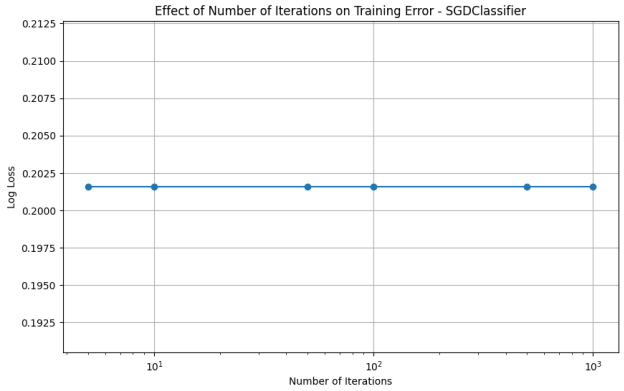


شکل ۲۱: Effect of Learning Rate on Training Error - SGDClassifier



شکل ۲۲: Effect of Number of Iterations on Training Error - SGDClassifie

مقادیر بهینه برای نرخ یادگیری و تعداد تکرار با توجه به نمودارها تعیین می‌شود.

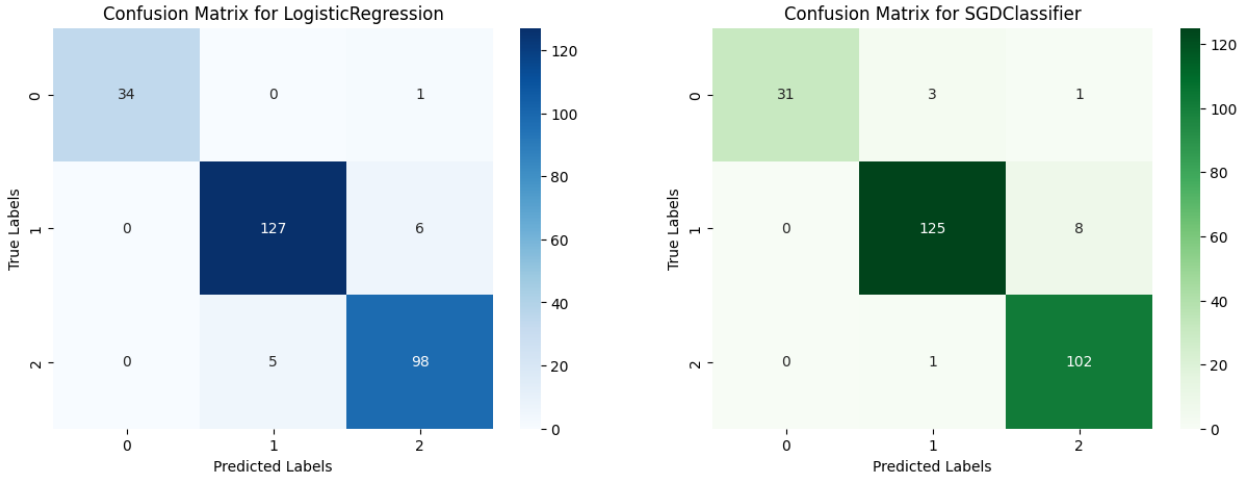


شکل ۲۳: Effect of Number of Iterations on Training Error - Logistic Regression

	precision	recall	f1-score	support
a	1.00	0.97	0.99	35
b	0.96	0.95	0.96	133
c	0.93	0.95	0.94	103
accuracy			0.96	271
macro avg	0.97	0.96	0.96	271
weighted avg	0.96	0.96	0.96	271

	precision	recall	f1-score	support
a	1.00	0.89	0.94	35
b	0.97	0.94	0.95	133
c	0.92	0.99	0.95	103
accuracy			0.95	271
macro avg	0.96	0.94	0.95	271
weighted avg	0.95	0.95	0.95	271

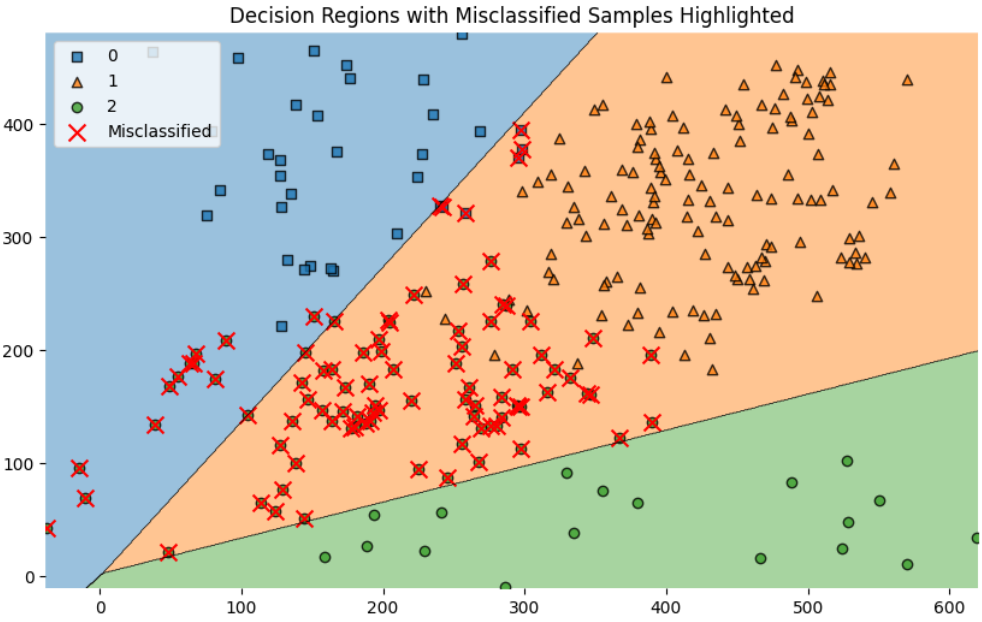
شکل ۲۴: نتایج پس از بهبود مدل



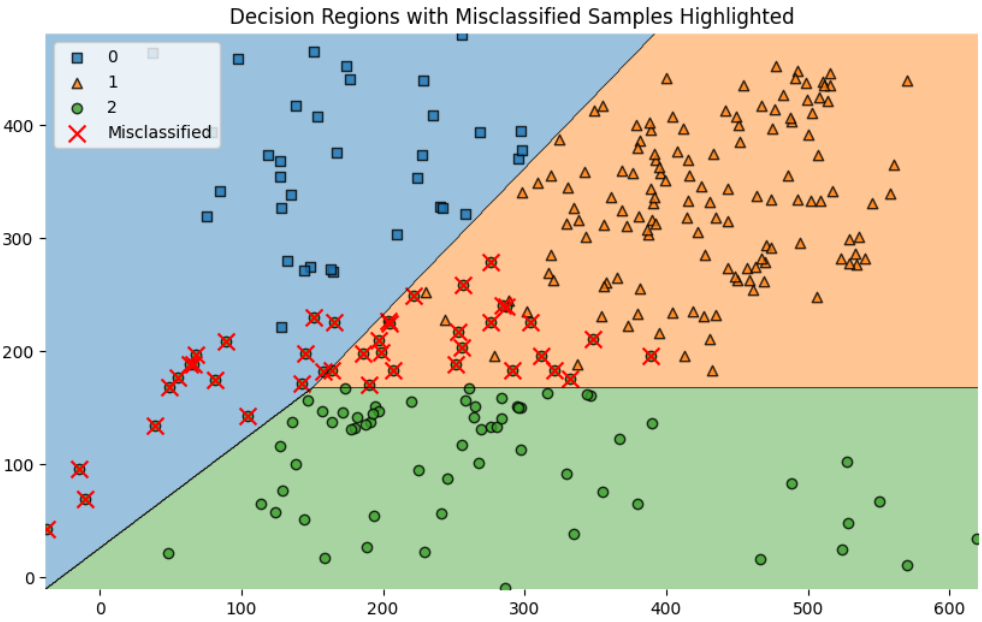
شکل ۲۵: مدل بهبود از پس Confusion Matrix

در این سری داده با توجه به پیچیدگی‌هایی که ایجاد کردیم، بهبود مدل به شکل قابل ملاحظه‌ای باعث تقویت نتایج و افزایش دقت شده است.

در نهایت به رسم نواحی تصمیم‌گیری می‌پردازیم.



شکل ۲۶: نواحی تصمیم‌گیری برای مدل LogisticRegression



شکل ۲۷: نواحی تصمیم‌گیری برای مدل SGDClassifier

۲ سوال دوم

۱.۲

مجموعه داده‌های مرکز داده‌های بلبرینگ دانشگاه (CWRU) منبع جامعی است که به طور گسترده در زمینه تشخیص نقص بلبرینگ استفاده می‌شود. این داده‌ها شامل اطلاعات سنسور ارتعاش برای بلبرینگ‌های عادی و معیوب است و به شرایط عادی، درایو آسیب‌دیده و فن آسیب‌دیده تفکیک می‌شود. داده‌ها با استفاده از حسگرهای ارتعاش در مکان‌های مختلف آزمایشگاهی جمع‌آوری می‌شوند و تحت شرایط عملیاتی مختلف ثبت می‌شوند.

محققان از این مجموعه داده برای توسعه مدل‌های یادگیری ماشینی مانند شبکه‌های عصبی کانولوشنی استفاده کرده‌اند تا نقایص بلبرینگ را با دقت بالا طبقه‌بندی و پیش‌بینی کنند. برای مثال، مدل FaultNet برای تجزیه و تحلیل داده‌های ارتعاشی و استفاده از تکنیک‌های پیشرفته پردازش سیگنال و یادگیری ماشینی به منظور دسته‌بندی انواع مختلف نقص‌های بلبرینگ طراحی شده است. از ویژگی های این سری داده می‌توان به موارد زیر اشاره کرد:

این مجموعه داده شامل طیف وسیعی از انواع نقص‌های بلبرینگ، از جمله آسیب‌های مسیر داخلی، مسیر خارجی و توپ‌ها است. داده‌ها تحت شرایط عملیاتی مختلف ضبط شده‌اند که شامل سطوح بار و سرعت‌های مختلف است. با جمع‌آوری داده‌های ارتعاشی با نرخ نمونه‌برداری بالا، این مجموعه داده اطلاعات دقیقی از رفتار دینامیکی بلبرینگ‌ها را فراهم می‌کند. این سری داده شامل بلبرینگ‌هایی با نقص‌های مصنوعی ایجاد شده در سطوح شدت مختلف است که در بسیاری از مطالعات استفاده شده است و همچنین برای توسعه و آزمایش مدل‌های یادگیری ماشینی استفاده می‌شود. از ویژگی‌های مهم دیگر آن این است که همراه با مستندات دقیق به طور رایگان قابل دانلود است.

مجموعه داده CWRU که در محیط آزمایشگاهی تولید شده، شامل داده‌های بلبرینگ‌ها استفاده شده در موتور القایی Reliance Electric با توان دو اسب بخار می‌باشد. این سیستم شامل یک ترانسدیوسر گشتاور، یک دینامومتر و واحد کنترلی است. داده‌ها انواع متفاوتی از خرابی‌ها را پوشش می‌دهند و به چهار دسته تقسیم شده‌اند: عادی در ۴۸ کیلوهرتز، عیب در سمت درایو در ۴۸ کیلوهرتز، عیب در سمت درایو در ۱۲ کیلوهرتز، و عیب در سمت فن در ۱۲ کیلوهرتز.

این دسته‌بندی‌ها شامل زیر مجموعه‌هایی برای شناسایی نوع خرابی‌ها هستند: - عیب بلبرینگ (B) - خرابی‌های داخلی - خرابی‌های خارجی، که بر اساس موقعیت نسبی نسبت به ناحیه باردهی دسته‌بندی شده‌اند: مرکزی (موقعیت ساعت ۶:۰۰)، عمودی (موقعیت ساعت ۳:۰۰)، و مخالف (موقعیت ساعت ۱۲:۰۰).

این عیوب با استفاده از فرآیند ماشینکاری الکترو-تخریبی (EDM) بر روی بلبرینگ‌های آزمایشی ایجاد شده‌اند و با قطرهای مختلف مشخص می‌شوند، مانند ۷، ۱۴، ۲۱، ۲۸ و ۴۰ میلی‌متر.

داده‌ها با دو فرکانس نمونه‌برداری مختلف، ۱۲ و ۴۸ کیلوهرتز جمع‌آوری شده‌اند، و اطلاعات ارتعاشی برای بارهای موتور در محدوده ۰ تا ۳ اسب بخار، در سرعت‌های موتوری بین ۱۷۲۰ تا ۱۷۹۷ دور در دقیقه (RPM) ثبت شده‌اند.

۲.۲

- آ) با توجه به باقی‌مانده شماره دانشجویی بر ۴، داده‌های Normal_0 و IR007_0 را دانلود می‌کنیم که از این سری داده‌ها، قسمت‌های X097_DE_time و X105_DE_time برای کلاس‌های نرمال و عیب استفاده می‌شوند.

برای ایجاد دیتای مورد نظر از M=۲۰۰ و N=۳۰۰ استفاده می‌کنیم و داده ها را زیر هم قرار می‌دهیم تا یه یک ماتریس $300 * 400$ برسیم. همچنین برای اینکه دیتای انتخاب شده در اجراهای مختلف تغییر نکند از یک random state استفاده می‌کنیم. کد مورد

استفاده برای ایجاد ماتریس با استفاده از داده خام به صورت زیر است.

```python
normal_bearings_data = pd.read_csv('/content/drive/MyDrive/ML2024/MP1/Q2/Normal.csv')
faulty_bearings_data = pd.read_csv('/content/drive/MyDrive/ML2024/MP1/Q2/Fault.csv')

# Function to select M samples of length N from the dataset
def select_samples(data, M, N, random_state=None):
    if random_state is not None:
        np.random.seed(random_state)

    samples = []
    for _ in range(M):
        start_index = np.random.randint(0, len(data) - N)
        sample = data[start_index:start_index + N]
        samples.append(sample)
    return np.array(samples)

M = 200
N = 300
normal_samples = select_samples(normal_bearings_data, M, N, 64)
faulty_samples = select_samples(faulty_bearings_data, M, N, 64)

labels_normal = np.zeros(M)
labels_faulty = np.ones(M)

normal_samples_flat = normal_samples.reshape(M, -1)
faulty_samples_flat = faulty_samples.reshape(M, -1)

data_combined = np.vstack((normal_samples_flat, faulty_samples_flat))

labels_combined = np.concatenate((labels_normal, labels_faulty))

df = pd.DataFrame(data_combined)
df['label'] = labels_combined
```

شکل ۲۸: کد تشکیل ماتریس از داده خام

	0	1	2	3	4	5	6	7	8	9	...	291	292	293	294	295	296	297	298	299	label
0	-0.184833	-0.159591	-0.107437	-0.046313	0.031084	0.071972	0.105351	0.100344	0.077188	0.027954	...	-0.040889	-0.007302	0.017941	0.015229	-0.024617	-0.070303	-0.099927	-0.097632	-0.085950	0.0
1	0.165849	0.162929	0.141650	0.101596	0.061124	0.066131	0.065922	0.067800	0.048399	0.033587	...	0.124752	0.131845	0.097841	0.027120	-0.013769	-0.029415	-0.023991	-0.021279	-0.034004	0.0
2	0.076979	0.113487	0.126421	0.098466	0.053614	0.018775	0.006676	0.016272	0.008345	0.002921	...	0.023991	0.070303	0.081151	0.054240	0.017524	-0.010639	-0.001252	0.015229	0.011057	0.0
3	0.079691	0.087618	0.060707	0.011474	-0.012100	0.010014	0.026911	0.035673	0.025034	0.003755	...	-0.041723	-0.014603	-0.015438	-0.004590	0.031084	0.084072	0.113278	0.120371	0.090956	0.0
4	0.015855	0.034422	0.058412	0.049859	0.042975	0.067591	0.098884	0.125795	0.103056	0.056743	...	-0.073224	-0.035673	0.015855	0.063002	0.102013	0.129342	0.141024	0.136434	0.091582	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	-0.016731	-0.147816	0.027127	0.152851	-0.014782	-0.110943	0.112568	0.057989	-0.150577	-0.101847	...	-0.102984	0.236668	0.251937	-0.159349	-0.092426	0.269480	0.088690	-0.005035	0.273541	1.0
396	-0.114517	-0.124588	-0.076994	0.113705	0.065786	-0.326982	-0.475123	-0.257785	0.580218	0.900378	...	-0.065461	0.124425	-0.038660	0.016244	0.101684	-0.073096	-0.158699	-0.111268	-0.155288	1.0
397	0.615467	-0.165034	-0.191186	0.718126	0.024040	-0.923119	-0.186151	0.528239	-0.168283	-0.504686	...	0.139207	0.082842	0.115329	0.037523	-0.170070	-0.005035	0.446209	0.459854	-0.078619	1.0
398	-0.174455	-0.323246	-0.169420	0.145867	0.139694	-0.115979	-0.018842	0.224485	0.127349	-0.104933	...	-0.076182	-0.131572	-0.262008	-0.091776	0.226597	0.242353	0.141156	0.075207	-0.026964	1.0
399	-0.156263	0.195572	0.130598	-0.264120	-0.111268	0.086903	-0.170070	-0.039959	0.307977	0.215876	...	-0.255023	0.077319	-0.096486	-0.267368	0.136283	0.110943	-0.179653	0.000975	0.166171	1.0

400 rows × 301 columns

شکل ۲۹: ماتریس تشکیل شده از داده خام

- ب) استخراج ویژگی‌ها یک مرحله حیاتی در یادگیری ماشینی است، به ویژه هنگام کار با داده‌های خام که به اشکال متفاوتی ظاهر می‌شوند. عملکرد یک مدل یادگیری ماشین به شدت به نمایش داده‌هایی که به آن تغذیه می‌شود بستگی دارد.

داده‌های خام می‌توانند دارای ابعاد بالا باشند و ممکن است حاوی مقدار زیادی اطلاعات غیرمرتبط باشند. استخراج ویژگی به

کاهش ابعاد داده‌ها کمک می‌کند با انتخاب تنها ویژگی‌های مرتبط‌تر، که می‌تواند کارایی و عملکرد الگوریتم یادگیری را بهبود ببخشد. ویژگی‌های خوب می‌توانند ماهیت داده‌ها را به خوبی به تصویر بکشند و الگوهای پنهان را برای مدل‌های یادگیری ماشینی واضح‌تر کنند. این می‌تواند منجر به دقت بهتر مدل شود. با استخراج و استفاده از یک مجموعه کوچکتر از ویژگی‌ها، الگوریتم‌های یادگیری ماشینی می‌توانند سریع‌تر اجرا شوند، که این موضوع به ویژه برای مجموعه داده‌های بزرگ و کاربردهای real-time مهم است. استخراج ویژگی می‌تواند به فیلتر کردن نویز از داده‌ها کمک کند. نویز می‌تواند الگوهای مفید را پوشانده و موجب عملکرد منفی مدل شود. ویژگی‌هایی که به خوبی انتخاب شده‌اند می‌توانند به قابل فهم بودن نتایج مدل کمک کنند. فضاهای ویژگی با ابعاد کمتر امکان تجسم داده‌ها را فراهم می‌کنند، که برای تجزیه و تحلیل داده‌ها و تفسیر آنها مهم است.

به طور خلاصه، استخراج ویژگی مؤثر منجر به نمایش داده‌هایی می‌شود که برای یادگیری ماشین مناسب‌تر است و احتمالاً اجازه می‌دهد مدل‌های ساده‌تر را به سرعت یاد بگیرند و بهتر عمل کنند.

برای استخراج ویژگی از هر ۱۴ روش موجود در جدول استفاده می‌کنیم که بر اساس آن داده به شکل زیر می‌شود.

	Standard Deviation	Peak	Skewness	Kurtosis	Crest Factor	Clearance Factor	Peak to Peak	Shape Factor	Impact Factor	Square Mean Root	Mean	Absolute Mean	Root Mean Square	Impulse Factor	label
0	0.082507	0.202148	0.043740	2.230742	2.432551	0.823773	0.386982	1.197809	2.913731	0.495372	0.009921	0.069378	0.083101	2.432551	0.0
1	0.075359	0.198810	-0.224888	2.644479	2.568763	0.854981	0.380723	1.227826	3.153993	0.482216	0.017637	0.063035	0.077395	2.568763	0.0
2	0.069840	0.172525	-0.342078	2.789945	2.397963	0.777674	0.338374	1.240234	2.974036	0.471007	0.017282	0.058010	0.071946	2.397963	0.0
3	0.078017	0.243871	-0.066986	3.197392	3.070616	1.049189	0.476060	1.253021	3.847546	0.482118	0.014867	0.063384	0.079421	3.070616	0.0
4	0.073519	0.178783	-0.151802	2.441625	2.402300	0.784316	0.334410	1.223575	2.939394	0.477439	0.011554	0.060823	0.074422	2.402300	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	0.320313	1.200071	0.148521	4.760211	3.742906	2.795766	2.278478	1.390491	5.204478	0.655169	0.014139	0.230584	0.320625	3.742906	1.0
396	0.287450	1.347075	0.268734	5.335045	4.683156	3.271152	2.293097	1.372203	6.426241	0.641720	0.010518	0.209621	0.287642	4.683156	1.0
397	0.271383	0.970063	-0.152105	4.965460	3.567543	2.454102	1.888958	1.381568	4.928803	0.628715	0.016973	0.196815	0.271913	3.567543	1.0
398	0.305948	1.309877	0.222683	5.730266	4.278733	3.153109	2.425806	1.413464	6.047834	0.644534	0.010735	0.216586	0.306137	4.278733	1.0
399	0.295095	1.423419	0.349173	5.902481	4.819104	3.447367	2.454395	1.400214	6.747777	0.642573	0.012755	0.210946	0.295370	4.819104	1.0

400 rows × 15 columns

شکل ۳۰: داده حاصل از استخراج ویژگی

- ج) شافل کردن یا مخلوط کردن، یک تکنیک در یادگیری ماشین است که برای رندوم کردن ترتیب نقاط داده در داخل یک مجموعه داده استفاده می‌شود. این گامی مهم در آماده‌سازی داده‌ها برای آموزش و ارزیابی مدل یادگیری ماشین است.

مجموعه داده‌ها ممکن است دارای یک ترتیب خاص باشند که می‌تواند منجر به تعصب ترتیب در آموزش شود. برای مثال، اگر داده‌ها بر اساس کلاس یا زمان مرتب شده باشند، مدل ممکن است به جای یادگیری الگوهای مرتبط با دیتا الگوهایی را یاد بگیرد که نتیجه ترتیب است. شافل کردن به کاهش این ریسک کمک می‌کند. با رندوم کردن ترتیب نقاط داده، شافل کردن کمک می‌کند تا اطمینان حاصل شود که مدل الگوهایی را یاد نمی‌گیرد که مختص به ترتیب داده‌ها است. این می‌تواند منجر به یک مدلی شود که بهتر به داده‌های دیده نشده تعمیم می‌یابد. هنگام انجام اعتبارسنجی متقابل، شافل کردن می‌تواند مهم باشد تا اطمینان حاصل شود که تاق‌ها به طور تصادفی ایجاد می‌شوند. این به دست آوردن یک تخمین قابل اعتماد از عملکرد مدل کمک می‌کند. در آموزش مینی‌بچ، مانند با استفاده از گرادیان کاهشی تصادفی، حیاتی است که هر مینی‌بچ نماینده کل مجموعه داده باشد. شافل کردن اطمینان می‌دهد که هر مینی‌بچ دارای ترکیبی از کلاس‌های مختلف است، که به به‌روزرسانی‌های گرادیان پایدار و معنادار کمک می‌کند. بدون شافل کردن، مدل ممکن است یک کلاس داده را در یک زمان ببیند و بیش از حد به آن کلاس خاص تنظیم شود. وقتی کلاس دیگری را می‌بیند، ممکن است بیش از حد در جهت دیگر تنظیم شود. شافل کردن به ارائه یک دید متعادل از کلاس‌ها کمک می‌کند. ارزش ذکر است که شافل کردن همیشه مناسب نیست. برای سری‌های زمانی یا دیگر داده‌های وابسته به توالی، شافل کردن ممکن است وابستگی‌های زمانی را که برای یادگیری مدل حیاتی هستند از بین ببرد.

به طور خلاصه، شافل کردن برای حفظ استحکام و تعمیم‌پذیری مدل‌های یادگیری ماشینی مهم است. این اطمینان می‌دهد که فرایند آموزش تحت تأثیر ترتیب داده‌ها نیست، که منجر به عملکرد بهتر مدل در داده‌های دیده نشده می‌شود. با این حال، باید با توجه به ماهیت داده‌ها و مسئله مورد نظر، به دقت از آن استفاده شود.

با استفاده از train_test_split همزمان داده را شافل و به دو دسته آموزش و ارزیابی با نسبت ۷۵ به ۲۵ تقسیم می‌کنیم.

```
from sklearn.model_selection import train_test_split
y = extracted_features_df['label']
X = extracted_features_df.drop('label',axis=1)
X_train, X_test, y_train, y_test = train_test_split(
  X,y , random_state=64,test_size=0.25, shuffle=True)
```

شکل ۳۱: شافل کردن و تقسیم به آموزش و ارزیابی

- د) نرمال‌سازی داده‌ها یک تکنیک پیش پردازش متداول در یادگیری ماشینی، شامل مقیاس‌بندی ویژگی‌های ورودی به یک محدوده مشابه است. این فرآیند از ایجاد بایاس برای برخی ویژگی‌ها بر سایر ویژگی‌ها در طول آموزش مدل جلوگیری می‌کند و منجر به یادگیری پایدارتر و موثرتر می‌شود. بسیاری از الگوریتم‌های یادگیری ماشین فرض می‌کنند که تمام ویژگی‌ها در یک مقیاس مشابه هستند. بدون نرمال‌سازی، یک ویژگی با محدوده گسترده ممکن است فرآیند یادگیری را تحت تأثیر قرار دهد و باعث شود مدل به آن ویژگی بیشتر توجه کند.

داده‌های نرمال‌سازی‌شده به بهبود سرعت همگرایی الگوریتم‌های بهینه‌سازی مبتنی بر گرادیان کمک می‌کنند. نرمال‌سازی اطمینان می‌دهد که هر ویژگی به طور تقریبی به میزان متناسبی به نتیجه نهایی کمک می‌کند. این مسئله در مواردی مهم است که ویژگی‌ها واحدها یا محدوده‌های مختلفی از مقادیر دارند.

روش‌های متداول نرمال‌سازی شامل موارد زیر هستند:

نرمال‌سازی Min-Max: داده‌ها را به یک محدوده ثابت، معمولاً از ۰ تا ۱، می‌رساند.

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

استانداردسازی (نرمال‌سازی Z-score): داده‌ها را به گونه‌ای تبدیل می‌کند که میانگین ۰ و انحراف معیار ۱ داشته باشند.

$$x_{\text{standard}} = \frac{x - \mu}{\sigma}$$

از نرمال‌سازی در بخش ارزیابی نیز استفاده می‌کنیم چراکه ثبات در پیش پردازش داده ها را بین مراحل آموزش و ارزیابی تضمین می‌کند. اگر داده‌های تست نرمال‌سازی نشوند، مدل ممکن است در طول استنتاج با توزیع ویژگی‌های غیرمنتظره مواجه شود که به طور بالقوه منجر به عملکرد کمتر از حد بهینه یا حتی پیش‌بینی‌های نادرست می‌شود. اگر نرمال‌سازی به طور برای داده‌های ارزیابی اعمال نشود، اختلاف حاصل در مقیاس ویژگی‌ها می‌تواند باعث تعصب یا عدم دقت در پیش بینی‌های مدل شود. نرمال‌سازی داده‌های ارزیابی بر اساس همان فاکتورهای مقیاس‌بندی که در داده‌های آموزشی استفاده شده تضمین می‌کند که مدل به خوبی به نمونه‌های دیده نشده تعمیم می‌یابد.

۳.۲

در این قسمت توضیح مختصری در رابطه با پیاده‌سازی مدل Logistic Regression انجام می‌دهیم و سپس به بررسی نتایج آن می‌پردازیم.

تابع فعال‌ساز سیگموئید: تابع sigmoid به عنوان تابع فعال‌ساز در خروجی مدل ما استفاده می‌شود. این تابع هر مقدار ورودی را به یک مقدار بین ۰ و ۱ نگاشت می‌کند، که برای طیقه‌بندی دو کلاسه مناسب است و خروجی را به عنوان احتمال تعلق یافتن ورودی به کلاس مثبت تفسیر می‌کند.

پیش‌بینی Logistic Regression: تابع logistic_regression با استفاده از ویژگی‌های ورودی x و بردار وزن ،w پیش‌بینی‌های y_hat را محاسبه می‌کند. ورودی x با بردار وزن w ضرب ماتریسی شده و از تابع سیگموئید عبور داده می‌شود تا پیش‌بینی‌ها ایجاد شوند.

تابع دو کلاسه کراس-انتروپی: تابع bce ،کراس-انتروپی را محاسبه می‌کند، که یک تابع زیان رایج برای مشکلات دسته‌بندی دو کلاسه است. این تابع اختلاف بین برچسب‌های واقعی y و احتمالات پیش‌بینی شده y_hat را اندازه‌گیری می‌کند.

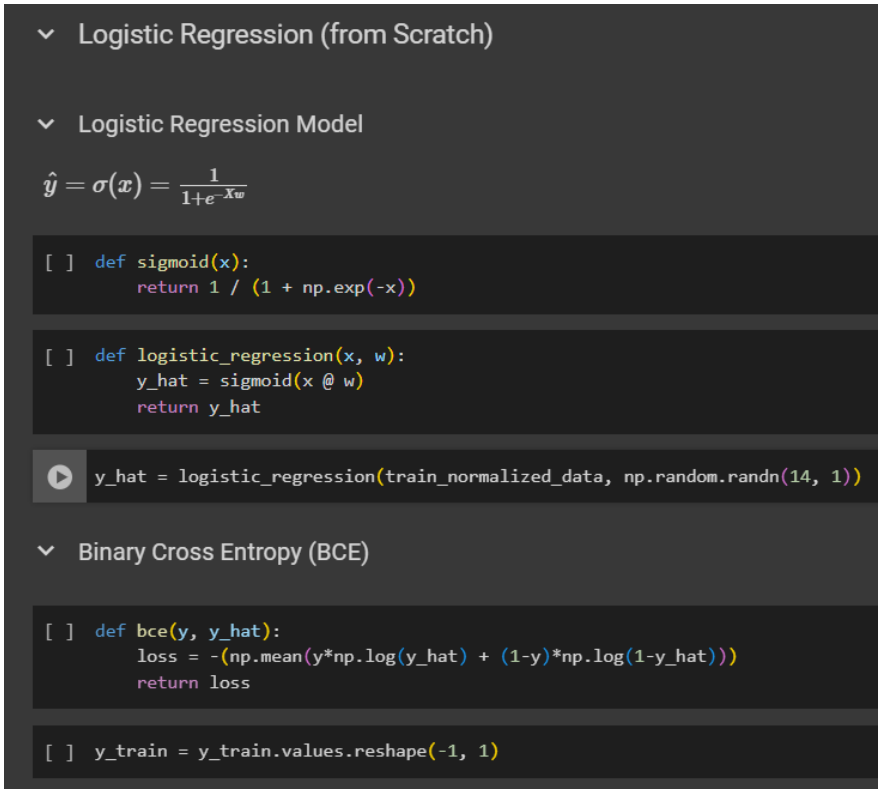
محاسبه گرادیان: تابع gradient ، گرادیان تابع خطا را نسبت به وزن‌های مدل محاسبه می‌کند.

به‌روزرسانی وزن‌ها با گرادیان نزولی: تابع gradient_descent ، وزن‌ها را با استفاده از نرخ یادگیری eta و گرادیان‌های محاسبه شده grads به‌روزرسانی می‌کند. نرخ یادگیری کنترل می‌کند که چقدر ما باید وزن‌ها را با توجه به گرادیان خطا تنظیم کنیم.

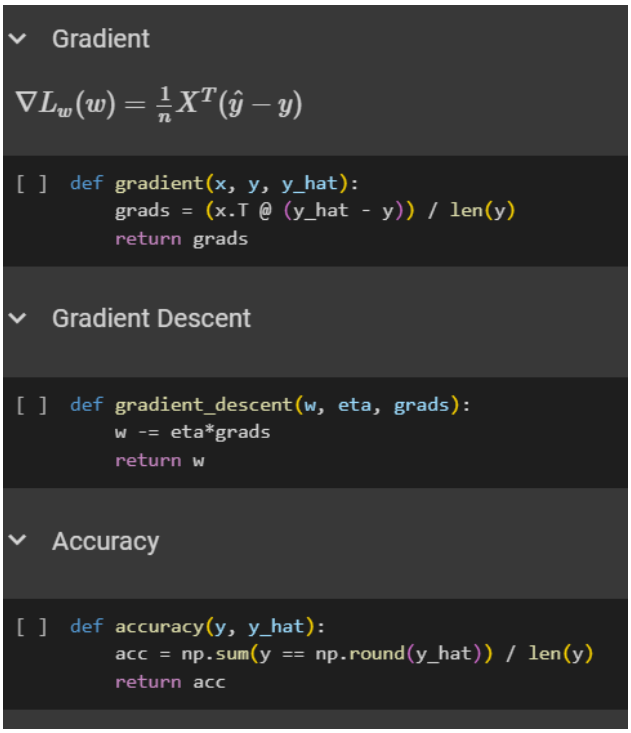
معیار دقت: تابع accuracy ، دقت طبقه‌بندی مدل را محاسبه می‌کند. این تابع پیش‌بینی‌های گرد شده y_hat را با برچسب‌های واقعی y مقایسه کرده و نسبت پیش‌بینی‌های صحیح را ارائه می‌دهد.

فرآیند آموزش: در طی حلقه آموزشی، برای تعداد دوره‌های ،n_epochs ما به طور مکرر پیش‌بینی‌ها را انجام می‌دهیم، خطا را محاسبه می‌کنیم، گرادیان‌ها را محاسبه می‌کنیم و وزن‌ها را به‌روزرسانی می‌کنیم.

کدهای مربوط به این قسمت به صورت زیر است.



شکل ۳۲: کد Logistic Regression



شکل ۳۳: کد Logistic Regression

```python
m = 13
w = np.random.randn(m+1, 1)
print(w.shape)

eta = 0.01
n_epochs = 2000
```

```python
error_hist = []

for epoch in range(n_epochs):
    # predictions
    y_hat = logistic_regression(train_normalized_data, w)

    # loss
    e = bce(y_train, y_hat)
    error_hist.append(e)

    # gradients
    grads = gradient(train_normalized_data, y_train, y_hat)

    # gradient descent
    w = gradient_descent(w, eta, grads)

    if (epoch+1) % 1 == 0:
        print(f'Epoch={epoch}, \t E={e:.4},\t w={w.T[0]}')
```

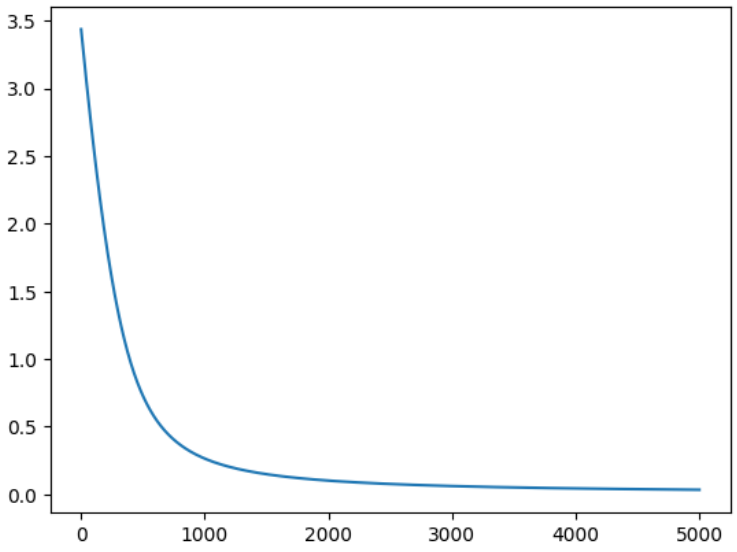
شکل ۳۴: کد Logistic Regression

در ادامه برای بررسی نتایج داده ارزیابی از ۴ معیار accuracy, precision, recall, F1 score استفاده می‌کنیم که کد مربوط به آن‌ها به صورت زیر است.

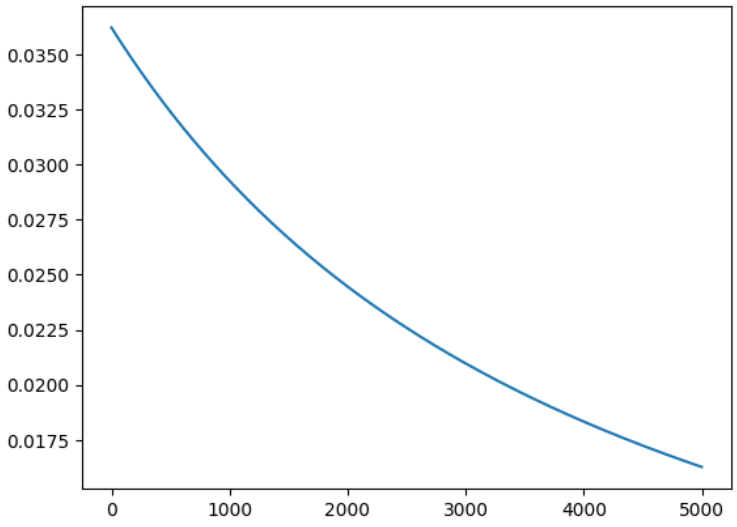
```python
# Calculate accuracy
def accuracy(y_true, y_pred):
    y_pred = np.round(y_pred)
    correct = sum(1 for true, pred in zip(y_true, y_pred) if true == pred)
    total = len(y_true)
    return correct / total

# Calculate precision
def precision(y_true, y_pred):
    y_pred = np.round(y_pred)
    true_positive = sum(1 for true, pred in zip(y_true, y_pred) if true == pred == 1)
    false_positive = sum(1 for true, pred in zip(y_true, y_pred) if true != pred == 1)
    return true_positive / (true_positive + false_positive) if (true_positive + false_positive) != 0 else 0

# Calculate recall
def recall(y_true, y_pred):
    y_pred = np.round(y_pred)
    true_positive = sum(1 for true, pred in zip(y_true, y_pred) if true == pred == 1)
    false_negative = sum(1 for true, pred in zip(y_true, y_pred) if true == 1 and pred == 0)
    return true_positive / (true_positive + false_negative) if (true_positive + false_negative) != 0 else 0

# Calculate F1 score
def f1_score(y_true, y_pred):
    y_pred = np.round(y_pred)
    prec = precision(y_true, y_pred)
    rec = recall(y_true, y_pred)
    return 2 * (prec * rec) / (prec + rec) if (prec + rec) != 0 else 0
```

شکل ۳۵: کد معیارهای ارزیابی متفاوت

با انجام مراحل آموزش و ارزیابی روی داده پیش‌پردازش شده در مراحل قبل به نتایج زیر می‌رسیم.



شکل ۳۶: نمودار تابع اتلاف داده آموزش



شکل ۳۷: نمودار تابع اتلاف داده ارزیابی

مشاهده می‌شود که نمودارهای توابع اتلاف در داده آموزش پس از ۱۰۰۰ تکرار و در داده تست در حدود ۵۰۰۰ تکرار به مقدار ثابت و بسیار کوچکی رسیده که نشان از رسیدن به مقدار نهایی و مطلوب آن است.

همچنین با بررسی توابع ارزیابی به نتایج Accuracy: 1.0| F1 Score: 1.0| Precision: 1.0| Recall: 1.0 می‌رسیم که نشان می‌دهد مدل به خوبی دو کلاس موجود در داده تست را تشخیص داده است.

به طور کلی نمودار اتلاف قبل از مرحله ارزیابی نمی‌تواند قطعیتی در رابطه با عملکرد مدل روی داده ارزیابی ارائه دهد و در صورت پیچیدگی یا الگوهای پنهان در داده امکان تفاوت عملکرد روی داده آموزش و ارزیابی وجود دارد. گاهی اوفات نیز با مسئله Overfitting

روبرو هستیم که در آن مدل به شکل زیاد از حد الگوهای موجود در داده آموزش را یاد گرفته است (گاها نویز یاد گرفته می‌شود) که این موجب عملکرد ضعیف در مواجهه با داده‌های دیده نشده می‌شود. در داده ما با توجه به تعداد زیاد داده و سادگی روابط موجود این عملکرد در رابطه با داده ارزیابی نیز مشابه داده آموزش بود. چندین راه‌حل متداول برای این مشکل وجود دارد. یکی از راه‌حل‌های موجود، تقسیم کردن داده به ۳ دسته است، دو دسته مانند قبل و یک دسته Validation این دسته در تنظیم پارامترهای مدل و انجام یک اعتبارسنجی پیش از مواجه شدن با داده‌های ارزیابی استفاده می‌شود.

راهکار دیگر استفاده از Cross Validation است، مبنای این روش تقسیم کردن داده آموزش به تعدادی دسته مشخص و انجام سه مرحله train, validation, test روی تمامی این دسته‌هاست.

۴.۲

در این قسمت با استفاده از یک مدل آماده در کتابخانه sklearn فرایند آموزش و ارزیابی را پیاده‌سازی می‌کنیم.

از دو مدل LogisticRegression و SGDClassifier برای این مسئله استفاده می‌کنیم. کدها و نتایج این پیاده‌سازی را به صورت زیر است.

```
model1 = LogisticRegression(solver='sag', max_iter=200, random_state=64)
model1.fit(train_normalized_data, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: Dat
  y = column_or_1d(y, warn=True)
```

```
LogisticRegression
LogisticRegression(max_iter=200, random_state=64, solver='sag')
```

```
[36] model.predict_proba(test_normalized_data)
```

```
[45] model1.score(train_normalized_data, y_train)
     1.0
```

```
[46] model1.score(test_normalized_data, y_test)
     1.0
```

شکل ۳۸: پیاده‌سازی و نتایج LogisticRegression

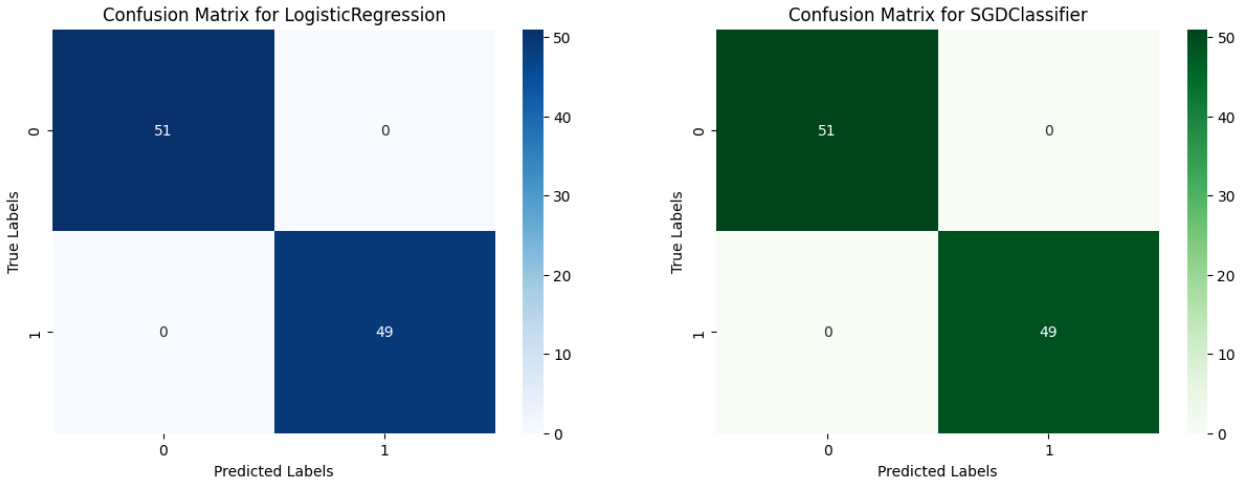
```
[54] model2 = SGDClassifier(loss='log_loss', random_state=64)
     model2.fit(train_normalized_data, y_train)

     /usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.
       y = column_or_1d(y, warn=True)
     SGDClassifier
     SGDClassifier(loss='log_loss', random_state=64)

[55] model2.score(train_normalized_data, y_train)

     1.0

     model2.score(test_normalized_data, y_test)

     1.0
```

شکل ۳۹: پیاده‌سازی و نتایج SGDClassifier

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	51
1.0	1.00	1.00	1.00	49
accuracy			1.00	100
macro avg	1.00	1.00	1.00	100
weighted avg	1.00	1.00	1.00	100

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	51
1.0	1.00	1.00	1.00	49
accuracy			1.00	100
macro avg	1.00	1.00	1.00	100
weighted avg	1.00	1.00	1.00	100

شکل ۴۰: نتایج داده ارزیابی با معیارهای مختلف



شکل ۴۱: Confusion Matrices

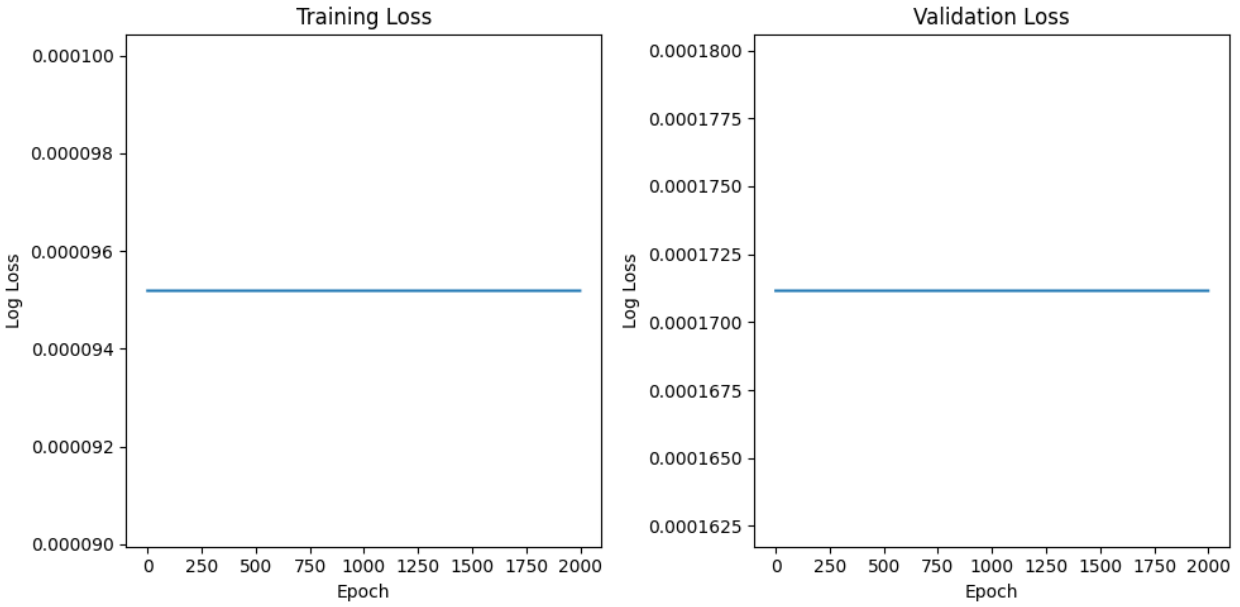
با توجه به نتایج مشخص است که این روش نیز به خوبی کلاس‌بندی را انجام می‌دهد و از نظر دقت تفاوتی با مدلی که در قسمت قبل کد زدیم ندارد.

برای رسم نمودار تابع اتلاف در این قسمت با استفاده از log loss به شکل زیر عمل می‌کنیم.

```python
train_losses = []
test_losses = []

n_epochs = 2000
for epoch in range(n_epochs):

    model2.fit(train_normalized_data, y_train)

    train_loss = log_loss(y_train, model2.predict_proba(train_normalized_data))
    test_loss = log_loss(y_test, model2.predict_proba(test_normalized_data))

    train_losses.append(train_loss)
    test_losses.append(test_loss)


plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(range(n_epochs), train_losses, label='Training Loss')
plt.title('Training Loss')
plt.xlabel('Epoch')
plt.ylabel('Log Loss')

plt.subplot(1, 2, 2)
plt.plot(range(n_epochs), test_losses, label='Validation Loss')
plt.title('Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Log Loss')

plt.tight_layout()
plt.show()
```

شکل ۴۲: کد رسم تابع اتلاف

نمودار تابع اتلاف مدل LogisticRegression به صورت زیر است.



شکل ۴۳: تابع اتلاف مدل LogisticRegression

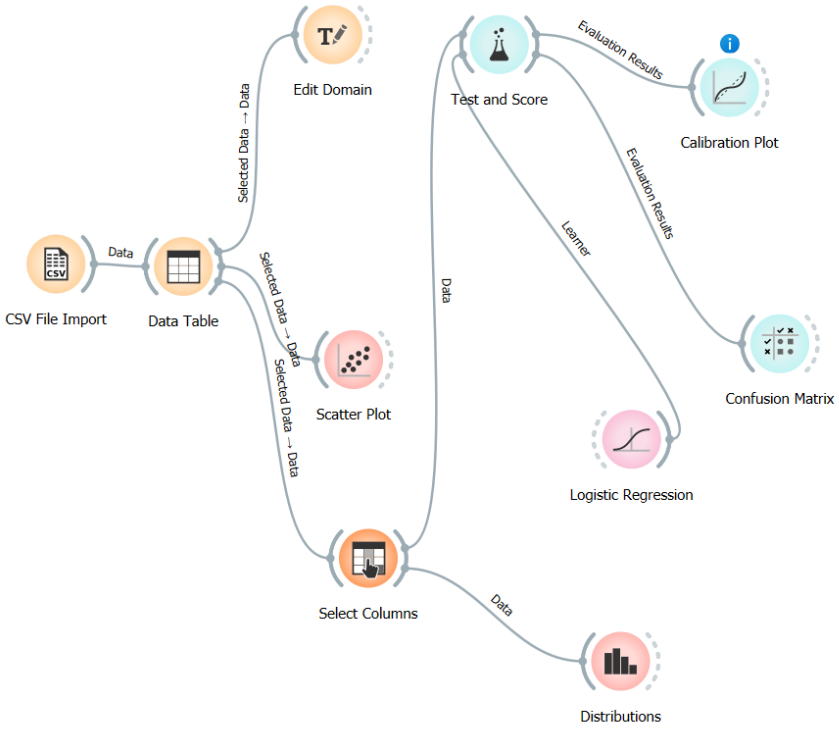
با توجه به نمودار تابع اتلاف می‌توان این نتیجه را گرفت که در این روش فرایند یادگیری سریع‌تر اتفاق می‌افتد و خطا نیز مقدار کمتری دارد.

۵.۲

Orange ابزاری است که کمک می‌کند بدون نیاز به کدنویسی، داده‌ها را تجزیه و تحلیل کرد. تنها با اتصال بلوک‌ها، به نام ویجت‌ها، برای انجام کارهایی مانند وارد کردن داده‌ها، آماده‌سازی داده‌ها، ساخت نمودارها، ساخت مدل‌ها و ارزیابی آن‌ها به سادگی قابل انجام است. با استفاده از Orange، می‌توان روندها را شناسایی کرد، روش‌های مختلف طبقه‌بندی داده‌ها را آزمایش کرد و پیش‌بینی انجام داد.

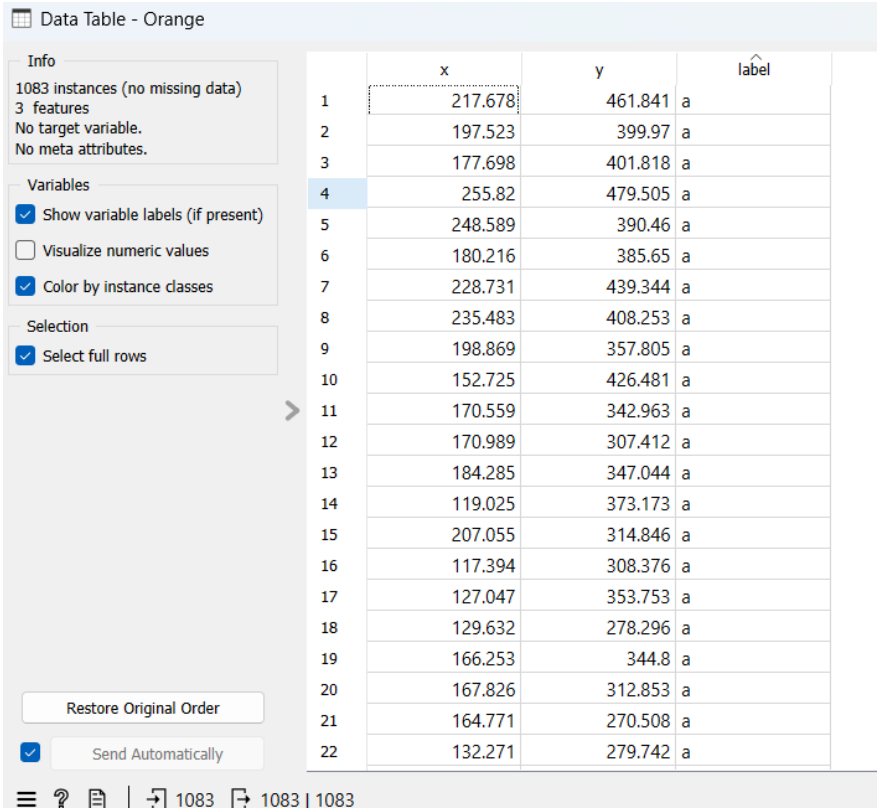
برای افرادی که با کد زنی آشنا نیستند یا تازه وارد فضای یادگیری ماشین شده‌اند خیلی آسان است و باعث می‌شود تجزیه و تحلیل داده‌ها جالب و سرگرم‌کننده باشد. در صورتی که توانایی کد زدن به زبان پایتون را داشته باشید، می‌توان از آن برای توسعه مدل‌ها و انجام کارهای بیشتر در Orange استفاده کرد.

در این قسمت قصد داریم تا با استفاده Orange، دیتایی که در قسمت آخر سوال۱ تولید کردیم را طبقه‌بندی کنیم تا تفاوت استفاده از این ابزار برایمان محسوس‌تر باشد.



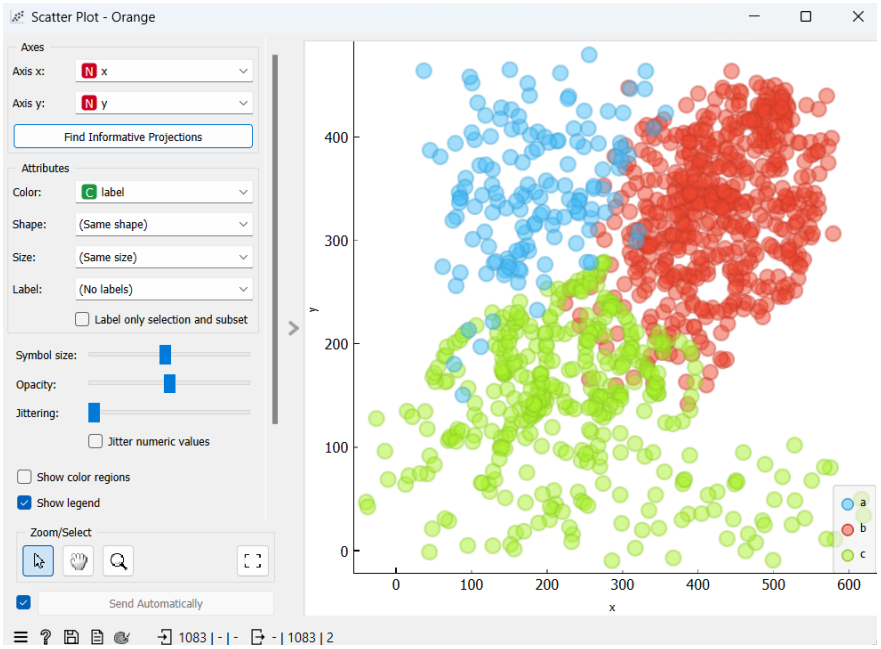
شکل ۴۴: مدل کامل ایجاد شده

در ابتدا با استفاده از CSV File Import داده را به نرم‌افزار می‌دهیم. سپس با استفاده از Data Table می‌توانیم داده را مشاهده کنیم. با استفاده از Scatter plot از کتابخانه Visulization می‌توان داده را رسم کرد. در صورت نیاز به تغییرات در داده می‌توان از Edit Domain استفاده کرد.

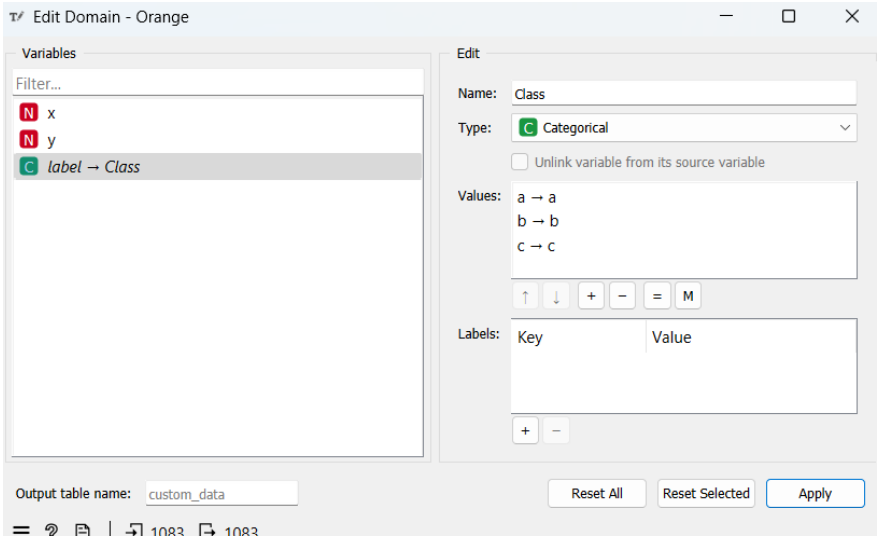


	x	y	label
1	217.678	461.841	a
2	197.523	399.97	a
3	177.698	401.818	a
4	255.82	479.505	a
5	248.589	390.46	a
6	180.216	385.65	a
7	228.731	439.344	a
8	235.483	408.253	a
9	198.869	357.805	a
10	152.725	426.481	a
11	170.559	342.963	a
12	170.989	307.412	a
13	184.285	347.044	a
14	119.025	373.173	a
15	207.055	314.846	a
16	117.394	308.376	a
17	127.047	353.753	a
18	129.632	278.296	a
19	166.253	344.8	a
20	167.826	312.853	a
21	164.771	270.508	a
22	132.271	279.742	a

شکل ۴۵: ماژول Data Table

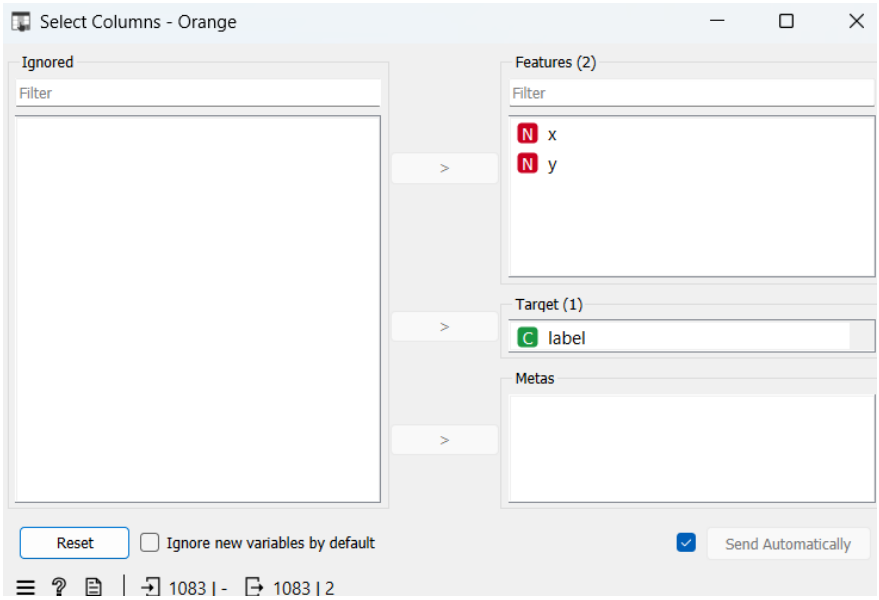


شکل ۴۶: ماژول Scatter Plot



شکل ۴۷: ماژول Edit Domain

برای مشخص کردن ستون داده برچسب در داده از Select Columns استفاده می‌کنیم. در این قسمت با انتخاب ستون برچسب داده، گزینه Label را انتخاب می‌کنیم.

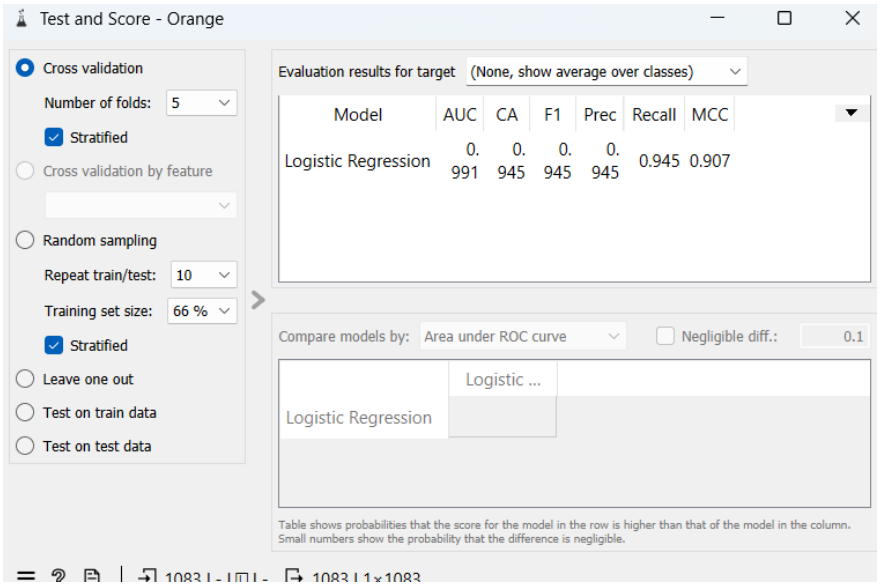


شکل ۴۸: ماژول Select Columns

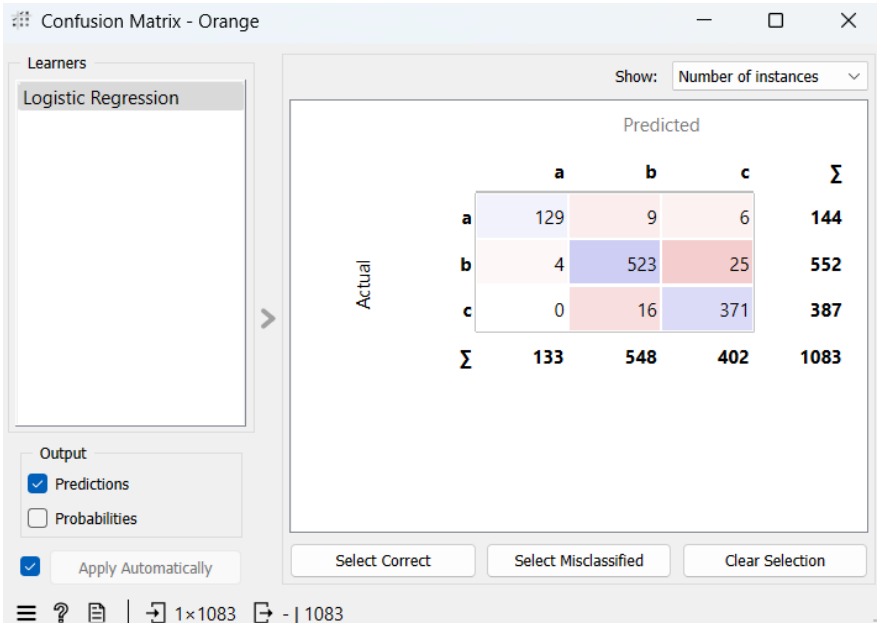
پس از آماده‌سازی داده، حال می‌توان از مدل برای آموزش استفاده کرد. با استفاده از مدل Logistic Regression از کتابخانه Model طبقه‌بندی را انجام خواهیم داد.

برای ارزیابی مدل از Test and Score از کتابخانه Evaluate استفاده می‌کنیم. این ماژول علاوه بر قابلیت ارزیابی بر اساس test و train ، قابلیت استفاده از روش Cross Validation را نیز دارد. همچنین این ماژول معیارهای ارزیابی متنوعی را اعم از Precision, Recall, Accuracy, ... در اختیار کاربر قرار می‌دهد. کتابخانه Evaluate ماژول‌های دیگری مثل Confusion Matrix, ROC, Performance

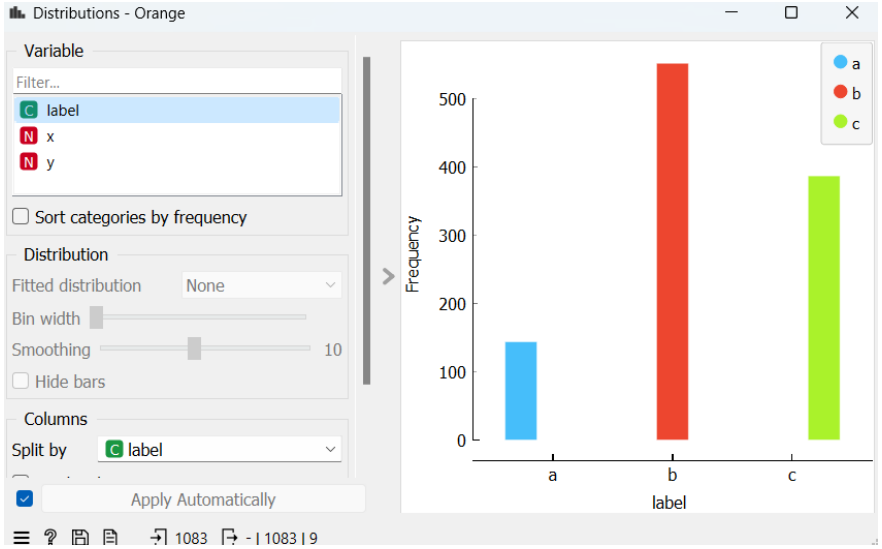
Curve, ... را نیز دارد.



شکل ۴۹: ماژول Test and Score



شکل ۵۰: ماژول Confusion Matrix



شکل ۵۱: ماژول Distribution

۳ سوال سوم

۱.۳

مجموعه داده‌ی szeged-weather که در کگل موجود است، داده‌های مرتبط با آب و هوا را از شهری در مجارستان، در بازه زمانی ۲۰۰۶ تا ۲۰۱۶ جمع‌آوری می‌کند. این مجموعه داده‌ی شامل اطلاعاتی در مورد دما است، مانند دما واقعی و دمایی که حس می‌شود و حداقل و حداکثر دماهای روزانه را نیز در خود دارد. علاوه بر این، مجموعه داده شامل رطوبت نسبی است که محتوای رطوبت در هوا را نشان می‌دهد، که برای تجزیه و تحلیل شاخص‌های آب و هوا مفید است. سرعت باد و جهت باد نیز ثبت شده‌اند که دید خوبی در مورد الگوهای باد ارائه می‌دهند. داده‌های میدان دید که در مجموعه داده وجود دارند، که می‌تواند برای بررسی چگونگی تاثیر شرایط آب و هوایی بر حمل و نقل و ایمنی مسیرها مورد مطالعه قرار بگیرد.

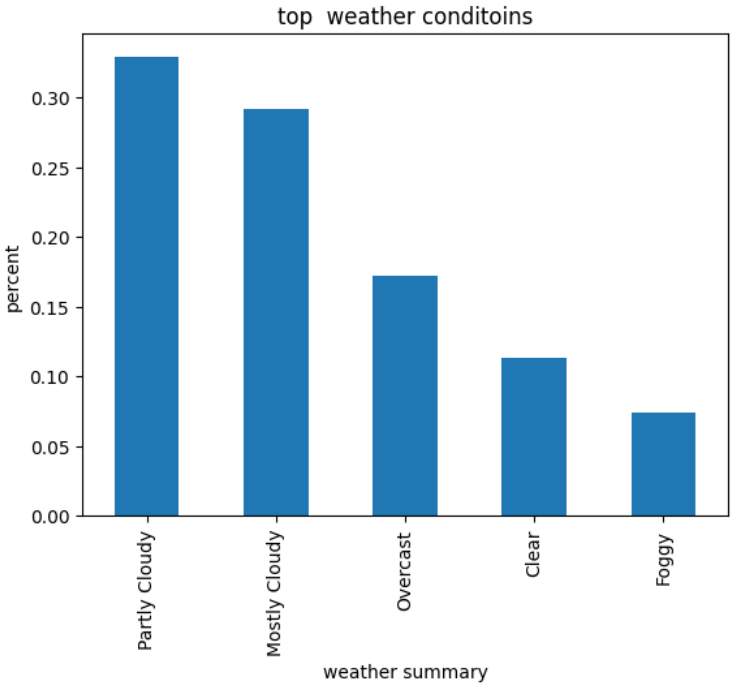
از دیگر مقادیر موجود در دیتا، فشار جوی است که برای مطالعه سیستم‌های آب و هوایی و پیش‌بینی تغییرات آب و هوا، یک جنبه حیاتی است. توصیف‌های دسته‌بندی شرایط آب و هوا، مانند صاف، ابری، بارانی یا مه‌آلود، نیز در این داده وجود دارد.

پس از مطالعه موارد موجود در این سری داده، به بررسی نمودارهای همبستگی و هیستوگرام پراکندگی می‌پردازیم.

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.
...	...	...	...	...	...	...	...	...	...	...	...	...
96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	31.0	16.1000	0.0	1014.36	Partly cloudy starting in the morning.
96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	20.0	15.5526	0.0	1015.16	Partly cloudy starting in the morning.
96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	22.038889	22.038889	0.56	8.9838	30.0	16.1000	0.0	1015.66	Partly cloudy starting in the morning.
96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	20.0	16.1000	0.0	1015.95	Partly cloudy starting in the morning.
96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	20.438889	20.438889	0.61	5.8765	39.0	15.5204	0.0	1016.16	Partly cloudy starting in the morning.

96453 rows × 12 columns

شکل ۵۲: شمای کلی مجموعه داده

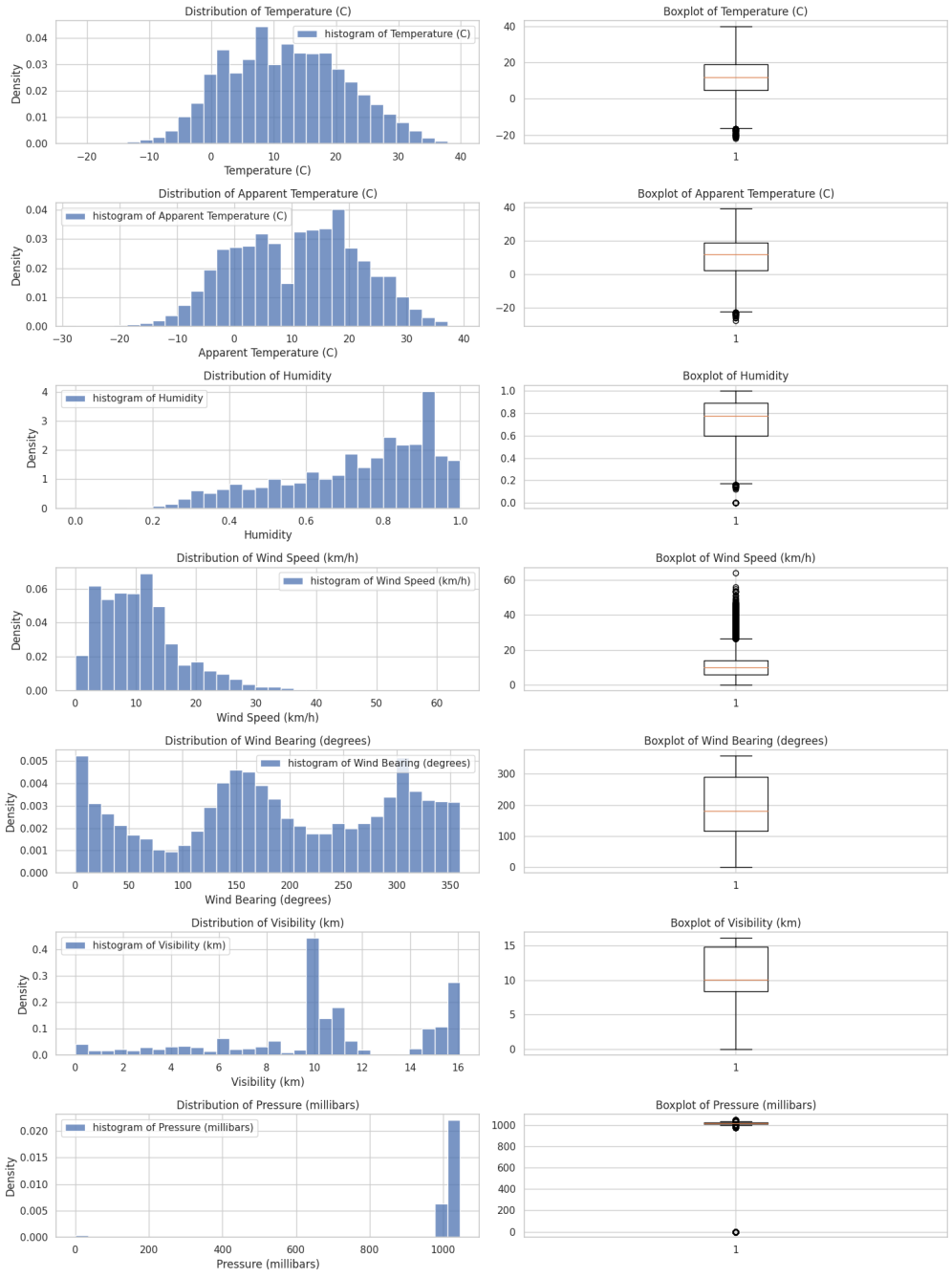


شکل ۵۳: وضعیت کلی مجموعه داده

Feature	Mean	Median	Variance	Standard Deviation	Range
Temperature (C)	11.93	12.00	91.23	9.55	61.73
Apparent Temperature (C)	10.86	12.00	114.42	10.70	67.06
Humidity	0.73	0.78	0.038	0.195	1.00
Wind Speed (km/h)	10.81	9.97	47.80	6.91	63.85
Wind Bearing (degrees)	187.51	180.00	11531.20	107.38	359.00
Visibility (km)	10.35	10.05	17.57	4.19	16.10
Pressure (millibars)	1003.24	1016.45	13681.96	116.97	1046.38

شکل ۵۴: اطلاعات آماری مجموعه داده

با توجه به هیستوگرام‌ها برای ویژگی‌های مهم تحلیل ارائه می‌دهیم.



شکل ۵۵: هیستوگرام ویژگی‌ها

دما: نمودار هیستوگرام دما نشان می‌دهد که داده‌ها بین −۲۰ تا ۴۰ درجه سلسیوس پراکنده‌اند و توزیع نرمالی دارند. این بدان معناست که اکثر داده‌های دما در اطراف میانگین تمرکز یافته‌اند.

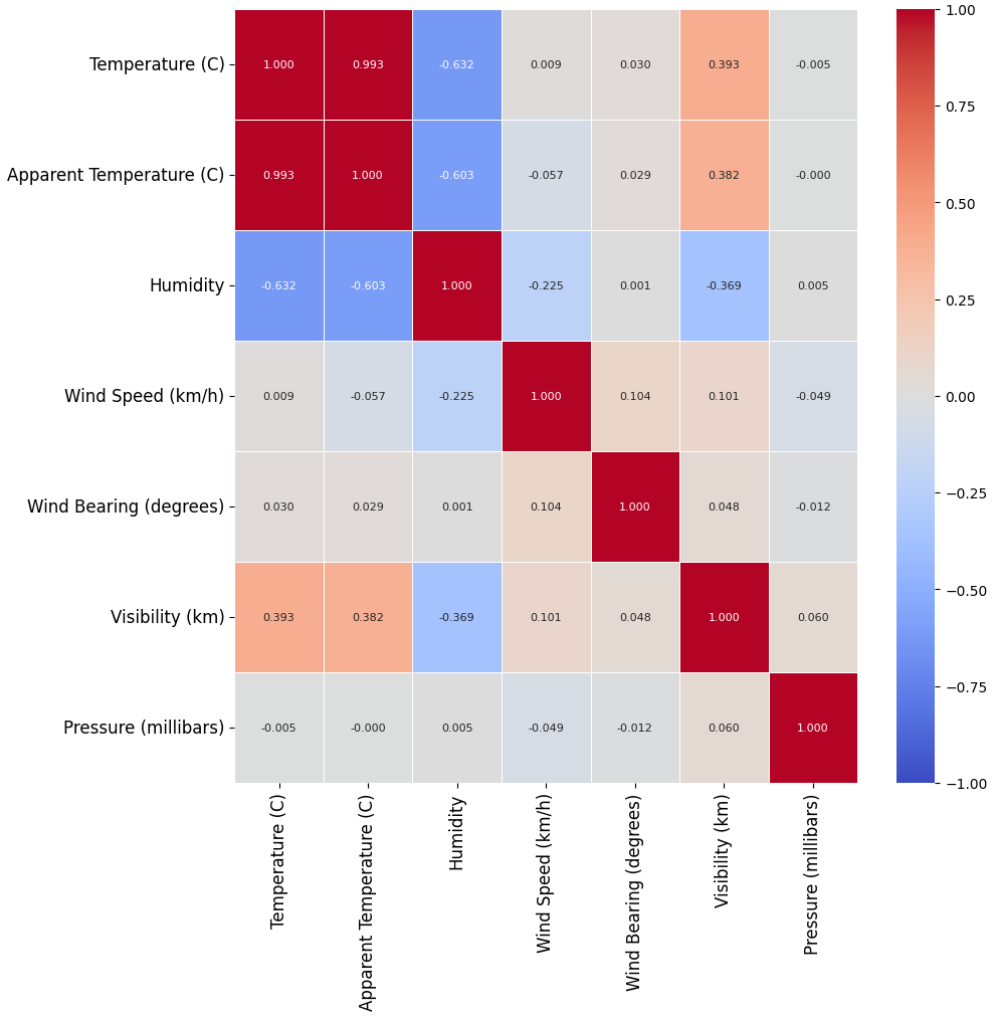
دمای ظاهری: دمای ظاهری نیز توزیعی نرمال دارد اما پراکندگی اندکی بیشتری نسبت به دمای واقعی دارد، که ممکن است به دلیل تأثیر عواملی مانند باد و رطوبت باشد.

رطوبت: هیستوگرام رطوبت نرمال نیست و نشان می‌دهد که Szeged شهری نسبتاً مرطوب است، با بیش از نیمی از مواقع رطوبتی

بالاتر از ۷۰ درصد.

سرعت باد: توزیع سرعت باد به سمت چپ کشیده شده و اغلب در محدوده ۳ تا ۱۷ کیلومتر بر ساعت قرار دارد، نشان می‌دهد که شهر بادگیر نیست.

داده فشار: این داده در محدوده محدودی قرار دارد و می‌توان گفت عموما ثابت است.



شکل ۵۶: ماتریس همبستگی

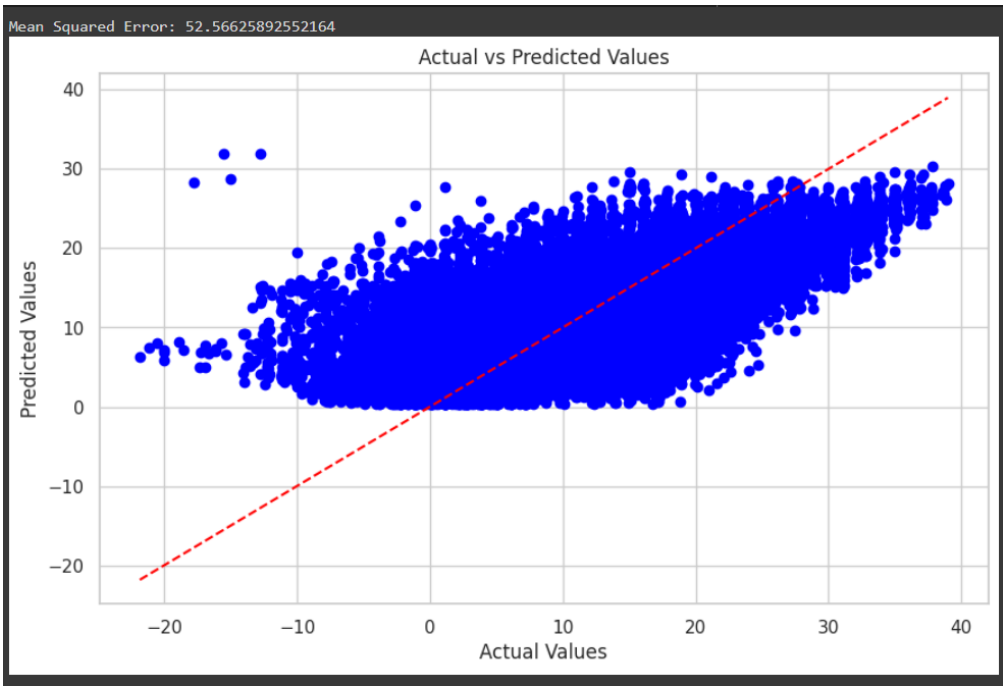
با توجه به ماتریس همبستگی، می‌توان روابط بین ویژگی‌ها را پیدا کرد. دما و دمای ظاهری رابطه خطی مستقیمی دارند، بیانگر این است که هر دو متغیر به طور هماهنگ تغییر می‌کنند. رطوبت با دما تقریبا رابطه عکس دارد و به نوعی افزایش رطوبت با کاهش دما همراه خواهد بود. ویژگی‌های باد و فشار همبستگی بسیار پایینی با دما دارند، در نتیجه برای پیش‌بینی دما کمک قابل ملاحظه‌ای به ما نخواهند کرد. ویژگی دیگری که همبستگی قابل توجهی با دما دارد، میدان دید است که تقریبا رابطی مستقیمی با دما دارد. برای انجام پیش‌بینی، انتخاب ویژگی‌هایی که همبستگی قوی‌تری با متغیر وابسته دارند، مانند استفاده از رطوبت و دید برای پیش‌بینی دمای، مفید خواهد بود.

۲.۳

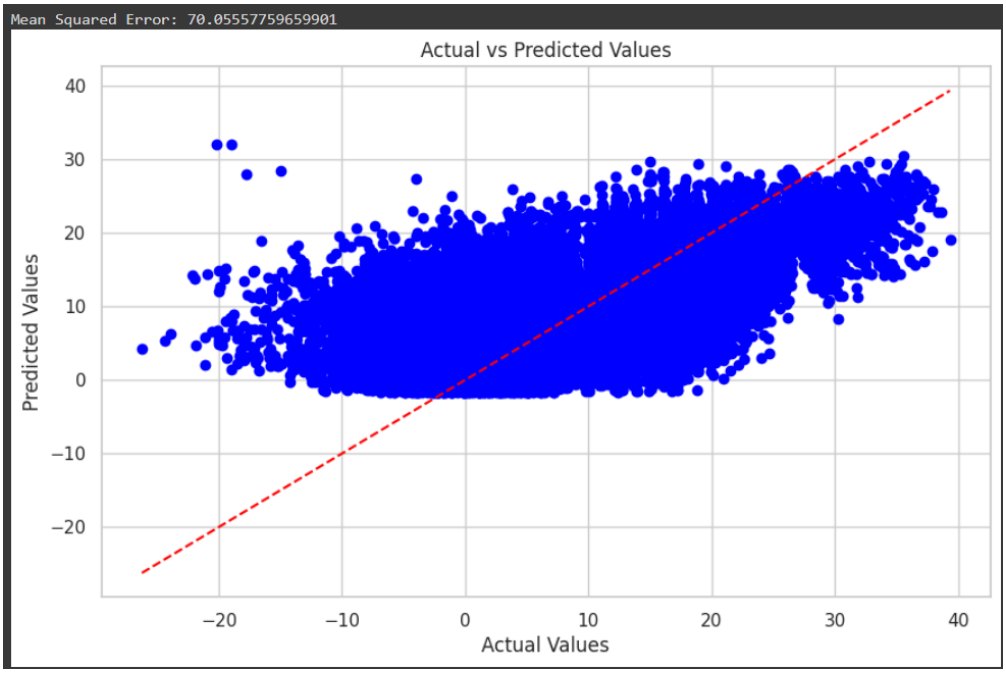
در این تمرین، ما قصد داریم تا با استفاده از روش‌های کمترین مربعات (LS) و کمترین مربعات بازگشتی (RLS)، به برازش یک مدل روی داده‌ها بپردازیم. RLS یک رویکرد تکامل یافته از LS است که اجازه می‌دهد مدل به صورت دوره‌ای و با دریافت داده‌های جدید به‌روزرسانی شود. هدف از این روش‌ها، محاسبه بهترین خط یا منحنی است که می‌تواند رابطه بین متغیرهای مستقل و وابسته را توضیح دهد، با تمرکز بر کمینه کردن تابع هزینه خطا که از طریق مربع خطاها محاسبه می‌شود. در روش LS، خطاهای بزرگ‌تر تأثیر بیشتری بر کمینه‌سازی تابع هزینه دارند زیرا از مربعات خطاها استفاده می‌شود.

در ادامه، پس از تقسیم داده به دو بخش تست و آموزش، ما با استفاده از معیار میانگین مربعات خطا (mean squared error)، کارایی تخمین‌های LS و RLS را ارزیابی خواهیم کرد. این رویکرد به ما کمک می‌کند تا بفهمیم چگونه مدل‌ها می‌توانند به طور موثری داده‌ها را یاد بگیرند دهند و چه میزان خطا در پیش‌بینی‌های خود دارند.

نتایج حاصل از استفاده از ویژگی‌های رطوبت و میدان دید برای تخمین دما و دما ظاهری به وسیله LS به ترتیب زیر است.



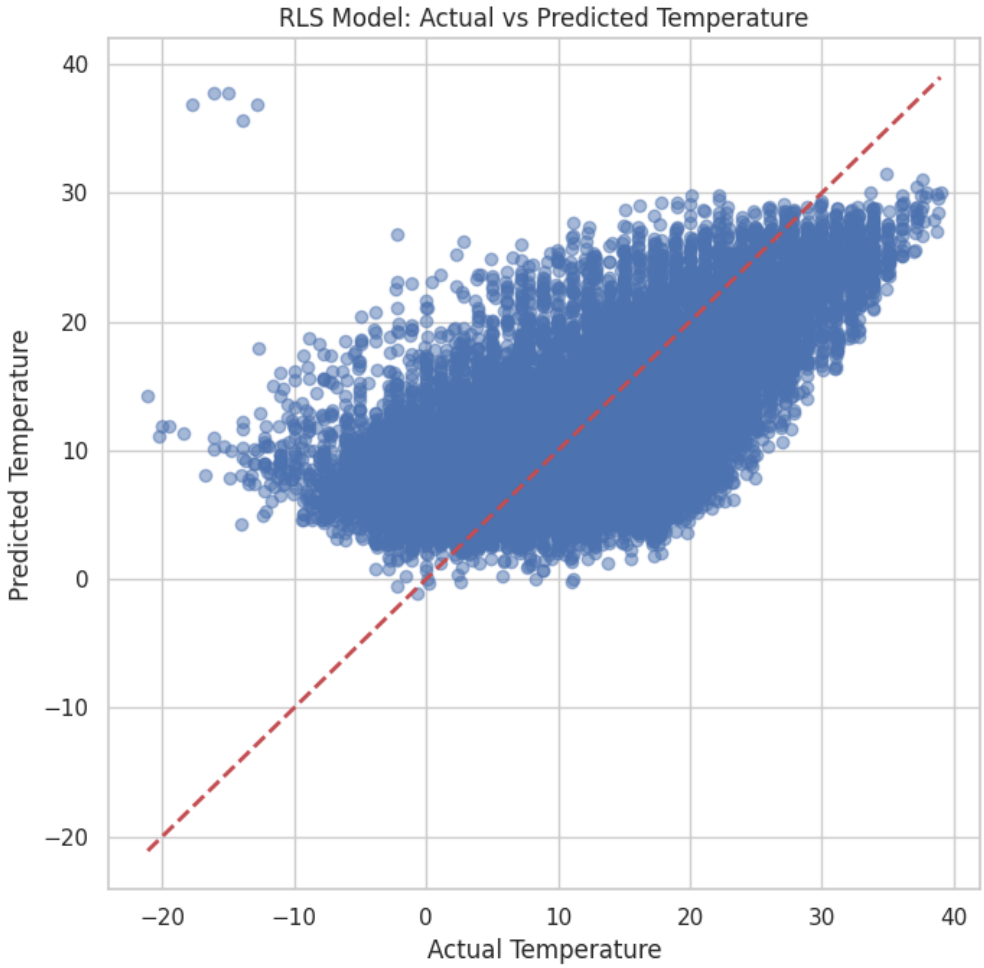
شکل ۵۷: تخمین دما



شکل ۵۸: تخمین دما ظاهری

مشاهده می‌شود که به علت توزیع فشرده داده‌ها تخمین مناسبی شکل نمی‌گیرد. با مقایسه دو تخمین می‌بینیم که تخمین دما خطای کمتری نسبت به دمای ظاهری دارد.

نتایج تخمین دما به وسیله RLS:

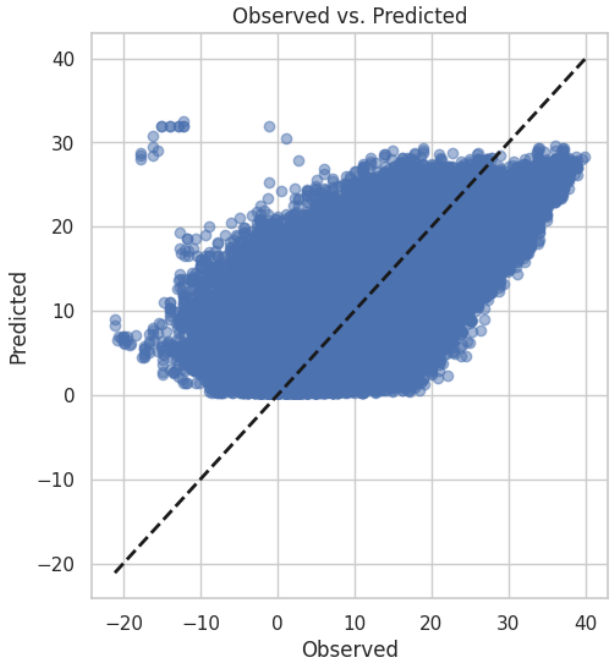


شکل ۵۹: تخمین دما با استفاده از RLS

۳.۳

روش کمترین مربعات وزنی (WLS) یک تکنیک پیشرفته است که بر مبنای روش کمترین مربعات (LS) ساخته شده، با این تمایز که در WLS هر نمونه داده با وزنی مشخص مورد بررسی قرار می‌گیرد. این فرآیند به‌خصوص در شرایطی مفید است که اطلاعات داده‌ای محدود یا با کیفیت متفاوت باشد. در WLS، هدف اصلی کاهش مجموع وزن‌دار مربع خطاها است، جایی که وزن هر خطا متناسب با وارونه‌ی واریانس خطای آن است. در نتیجه، داده‌هایی که نویز کمتری دارند و قابل اعتمادتر هستند، تأثیر بیشتری در تعیین پارامترهای مدل خواهند داشت. به این ترتیب، WLS با دادن وزن کمتر به داده‌های با نویز بیشتر، اطمینان حاصل می‌کند که نتایج تخمین بیشتر توسط نمونه‌های قابل اعتماد شکل گرفته است.

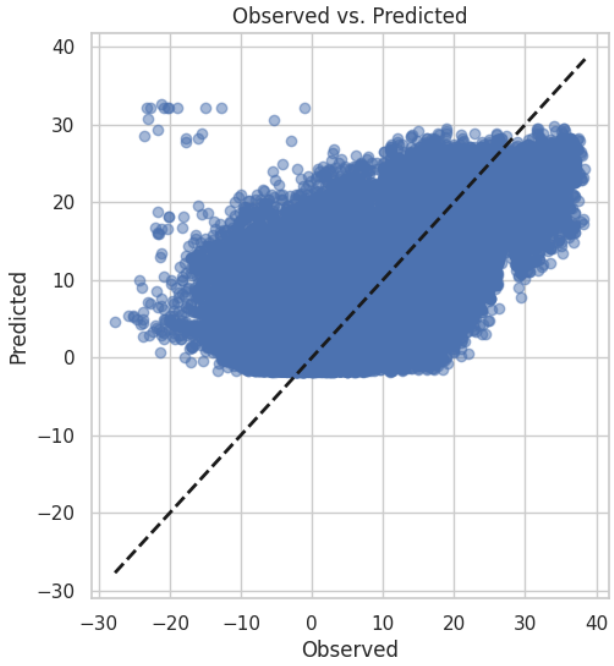
در این قسمت نیز مانند قسمت قبل تخمین دما و دمای ظاهری را انجام می‌دهیم.



شکل ۶۰: تخمین دما

```
Mean Squared Error (MSE) on the test set: 52.563
                            WLS Regression Results
==============================================================================
Dep. Variable:        Temperature (C)   R-squared:                       0.429
Model:                            WLS   Adj. R-squared:                  0.429
Method:                 Least Squares   F-statistic:                 2.715e+04
Date:                Sun, 07 Apr 2024   Prob (F-statistic):               0.00
Time:                        20:55:25   Log-Likelihood:            -2.4550e+05
No. Observations:               72339   AIC:                         4.910e+05
Df Residuals:                   72336   BIC:                         4.910e+05
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const             27.7125      0.152    181.923      0.000      27.414      28.011
Humidity         -27.4551      0.147   -186.159      0.000     -27.744     -27.166
Visibility (km)    0.4251      0.007     61.862      0.000       0.412       0.439
==============================================================================
Omnibus:                     2052.552   Durbin-Watson:                   1.997
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2231.596
Skew:                          -0.423   Prob(JB):                         0.00
Kurtosis:                       3.158   Cond. No.                         86.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

شکل ۶۱: پارامترهای مدل برای تخمین دما



شکل ۶۲: تخمین دمای ظاهری

```
Mean Squared Error (MSE) on the test set: 70.051
                            WLS Regression Results
==============================================================================
Dep. Variable:     Apparent Temperature (C)   R-squared:                       0.392
Model:                                 WLS   Adj. R-squared:                  0.392
Method:                      Least Squares   F-statistic:                 2.330e+04
Date:                     Sun, 07 Apr 2024   Prob (F-statistic):               0.00
Time:                             21:06:35   Log-Likelihood:            -2.5597e+05
No. Observations:                    72339   AIC:                         5.119e+05
Df Residuals:                        72336   BIC:                         5.120e+05
Df Model:                                2
Covariance Type:                 nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const             27.3315      0.176    155.252      0.000      26.986      27.677
Humidity         -29.1152      0.170   -170.822      0.000      -29.449     -28.781
Visibility (km)    0.4760      0.008     59.937      0.000       0.460       0.492
==============================================================================
Omnibus:                     3213.586   Durbin-Watson:                   1.997
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3663.026
Skew:                          -0.535   Prob(JB):                         0.00
Kurtosis:                       3.264   Cond. No.                         86.3
==============================================================================
```

شکل ۶۳: پارامترهای مدل برای تخمین دمای ظاهری

۴.۳

این بخش به معرفی و بررسی الگوریتم‌های حداقل مربعات بازگشتی (RLS) می‌پردازد که در زمینه‌های پردازش سیگنال و فیلترهای تطبیقی کاربرد دارند. تمرکز این بررسی بر استفاده از فرآیند تجزیه QR است که برای ساده‌سازی ماتریس‌های داده‌ای و بهبود کارایی RLS به کار می‌رود.

این روش، بر بهبود پایداری عددی الگوریتم RLS تحت تأثیر کوانتیزاسیون متمرکز است و نشان می‌دهد چگونه با استفاده از ساختارهای آرایه‌ای سیستولیک می‌توان پردازش‌های موازی را بهبود بخشید. از ویژگی‌های کلیدی ،RLS تطبیق و به‌روزرسانی دوره‌ای ضرایب فیلتر بر اساس داده‌های جدید برای کمینه کردن خطا است. همچنین، این روش به تحلیل دقت، پایداری و ملاحظات محدوده دینامیکی می‌پردازد که از اهمیت بالایی در کاربردهای عملی برخوردار هستند. در نهایت، استفاده از مثال‌های عملی برای نشان دادن کاربرد الگوریتم در بهینه‌سازی مسائل مختلف نیز مورد تأکید قرار می‌گیرد.