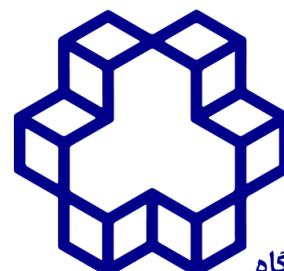


بسم الله الرحمن الرحيم



دانشگاه
خواجہ نصیرالدین طوسی
K. N. Toosi University
of Technology

پاسخ مینی پروژه اول یادگیری ماشین

Google Colab
GitHub

نگارش: علی شعبانپور مقدم - هدیه شوشیان

شماره دانشجویی: ۴۰۲۰۷۳۰۴-۴۰۳۰۸۰۵۴

استاد درس: دکتر مهدی علیاری شوره دلی

فهرست مطالب

۱	پیش بینی آب و هوا با یادگیری ماشین	
۳		دادگان
۳		۱.۱
۳		۱.۱.۱
۳		۲.۱.۱
۳		۳.۱.۱
۵		۴.۱.۱
۶	آموزش مدل	۲.۱
۶		۱.۲.۱
۷		۳.۱
۸		۱.۳.۱
۱۰		Extra ۴.۱
۱۱	تشخیص عیب یاتاقان غلتی بر مبنای دست هبندی های سلسله مراتبی	۲
۱۱		دادگان ۱.۲
۱۱		۱.۱.۲
۱۳		۲.۱.۲
۱۴		۳.۱.۲
۱۴	پیش پردازش و استخراج ویژگی	۲.۲
۱۴		۱.۲.۲
۱۶		۲.۲.۲
۱۹	آموزش مدل	۳.۲
۱۹		۱.۳.۲
۲۰		۲.۳.۲
۲۷		۳.۳.۲
۲۸		۴.۳.۲
۲۹	محصول	۴.۲



۱ پيش بيني آب و هوا با يادگيري ماشين

۱.۱ دادگان

۱.۱.۱

در اين مقاله چهار مكان از موريس از نظر آب و هوا مورد بررسی قرار گرفته است. پaramترهای آب و هوایی مورد استفاده در این مقاله شامل دما، سرعت باد، جهت باد، فشار، رطوبت و میزان ابری بودن است. داده های مورد استفاده از چندین منبع مختلف جمع آوری شده اند از جمله ایستگاه های هواشناسی محلی و پيش بيني جهانی. داده های ایستگاه هواشناسی محلی از چهار ایستگاه های محلی بطور همزمان جمع آوری شده اند و ويژگی آنها اين است که داده هایي دقیق بر حسب ساعت یا روز ارائه می دهند. در حالی که داده های جهانی از مدل های پيش بيني آب و هوا از قبیل *ERAS, GFS* استفاده می کند و اطلاعات آب و هوایی چند روز آینده را با دقت کمتری ارائه می کند.

پس از جمع آوری داده ها برای تشکیل دیتابست ابتدا یکپارچه سازی بر حسب ساعت صورت گرفته است بطوری که اگر فرمت زمان بندی بصورت ساعتی باشد و برای داده های گمشده از میانگین محلی برای جایگزین بهره برده شده است. و دیتابست نهایی ترکیبی از دیتابهای ایستگاه ها و پيش بيني جهانی است.

۲.۱.۱

در دادگان کل شهرهای BASEL , BUDAPEST , DE BILT , DRESDEN , DUSSELDORF , HEATHROW , KASSEL , MAASTRICHT , MALMO , MONTELIMAR , MUENCHEN , OSLO , PERPIGNAN , ROMA LJUBLJANA موجود است که از این بین سه شهر SONNBLICK , STOCKHOLM , TOURS Montelimar, Perpignan and Tours تنها از شهرهای فرانسه می باشد.

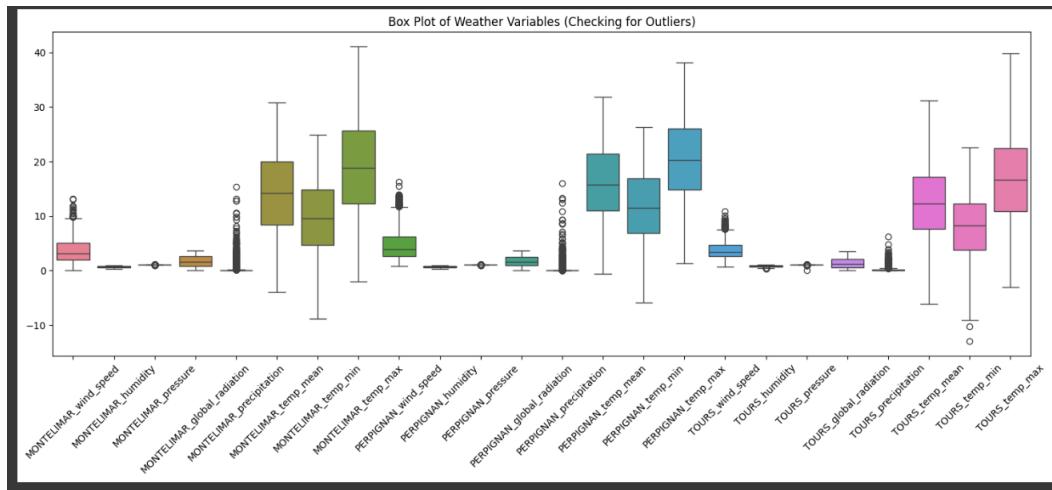
توضیح کد: ابتدا تمامی کتابخانه ها را فراخوانی می کنیم و سپس با دستور *kagglehub.dataset.download* دیتابست را دانلود کرده و پس از ذخیره در آن را پرینت می کنیم. سپس ستون هایی از دیتابست را که شامل تاریخ و ماه و بیان شده ای آب و هوایی شهرهای فرانسه هستند را در *df* ذخیره می کنیم.

۳.۱.۱

در این دادگان 3654 نمونه به ازای همین تعداد روز از تاریخ 01/01/2000 تا تاریخ 01/01/2010 وجود دارد. توضیح کد: با دستور *pd.to_datetime* فرمت عددی تاریخ را به فرمت زمانی یا آرایه ای تبدیل می کنیم. و با دستور (*df[“DATE”].min()*) مقدار مینیمم تاریخ و ماکزیمم آن که نمایانگر زمان شروع و پایان نمونه برداری است را بدست می آوریم.

در این مقاله پيش پردازش های – *MissingDataHandling, Min – MaxNormalization, TimeAlignmentorSynchronization, FeatureExtraction, Time – BasedSampling*

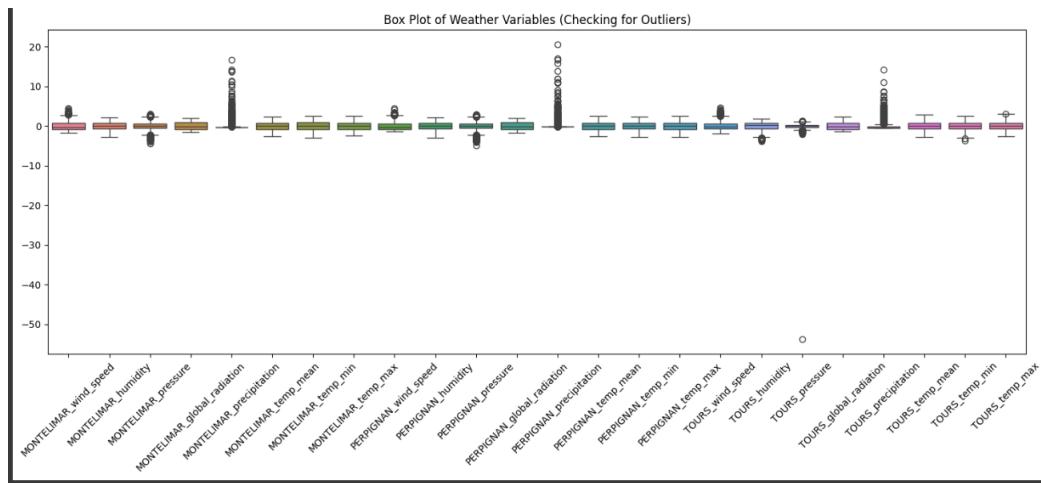
انجام شده است. در دادگان ما دیتابی از دست نرفته است پس نیاز به جایگزین داده های گمشده نمی باشد. اما جنس داده ها از نوع رطوبت و دما و... می باشند پس نیاز به نرمال سازی است. هماننگ سازی داده ها و نمونه برداری زمانی نیز در دادگان از قبل لحاظ شده است. استخراج ویژگی نیز انجام لحاظ می گردد. پس کافی است نرمال سازی و استخراج ویژگی صورت گیرد. ما در این سوال به منظور درک بهتر استاندارد سازی را نیز انجام دادیم. پیش از انجام پيش پردازش ها با کس پلات داده های سه شهر فرانسه به ازای هشت ویژگی به شکل زیر بدست آمده است. مشاهده می کنیم به علت تفاوت جنس داده ها و وجود داده های پرت و تراکم داده ها در مقادیر مشخصی می توانیم از استاندارد سازی و نرمالیزه سازی بهره ببریم.



شکل ۱: باکس پلات برای داده بدون پیش پردازش

پس از استاندارد سازی دادگان به شکل زیر خواهند شد. استاندارد سازی براساس میانگین و واریانس از فرمول زیر محاسبه گردید.

$$Z = \frac{(X - \mu)}{\sigma}$$



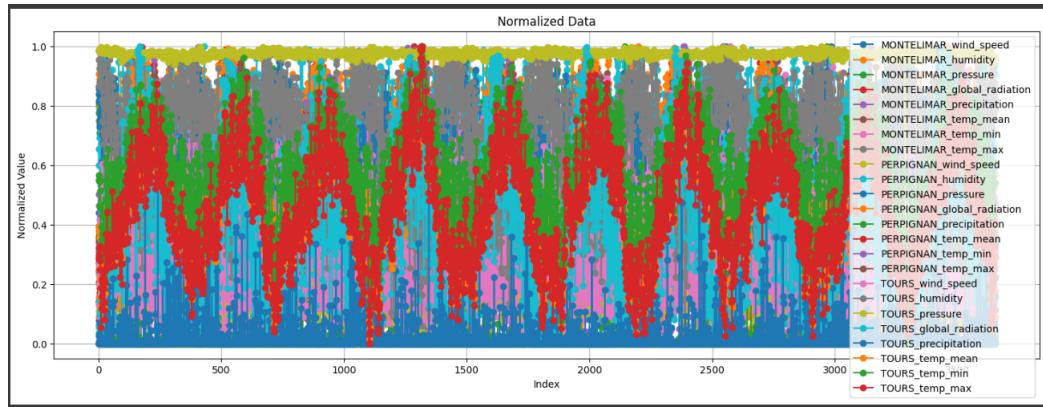
شکل ۲: باکس پلات برای داده پس از استاندارسازی

توضیح کد: با اضافه کردن کتابخانه `importStandardScaler` و دستور `sklearn.preprocessing` استاندارسازی تعریف میگردد کافی است ستون های عددی `df` را با دستور `df standard.selectdtypes(include = ["number"]).columns` جدا کرده و ستون تاریخ را که به شکل آرایه تبدیل شده است حذف کنیم. در نهایت با دستور `scaler.fittransform` دیتاهای استاندارد سازی شده و با نام `dfstandard` ذخیره می گردد. حال نرم افزاری داده ها را انجام می دهیم که در مقاله نیز انجام شده است. بطوری که مقادیر دیتا ها اعدادی بین صفر تا یک به خود می گیرند و از فرمول

زیر محاسبه می شود:

$$X_{norm} = \frac{(X - X_{min})}{(X_{max} - X_{min})}$$

و خروجی به شکل زیر خواهد بود:



شکل ۳: باکس پلات برای داده پس از استاندارسازی

توضیح کد: با اضافه کردن کتابخانه `sklearn.preprocessing` و با استفاده از دستور `MinMaxScaler` نرمال سازی تعریف می گردد. حال مانند حالت قبل کافی است این دستور را روی ستون ها عددی اعمال کنیم و ستون ماه که از نوع عددی نیست و ازنوع آرایه هست را جدا در نظر بگیریم. و در نهایت با دستور `dfnormal = dftrain[['temp_mean', 'temp_min', 'temp_max', 'humidity', 'wind_speed', 'precipitation', 'global_radiation']].dropna().values` دادگان نرمال سازی شده را در `scaler.fittransform` ذخیره نماییم.

۴.۱.۱

صورت سوال از ما خواسته داده ها را به نمونه هایی پنجره پنج تابی که با یکدیگر همپوشانی چهار دارند تقسیم نماییم. همچنین داده ها فقط برای سال 2009 در نظر گرفته شوند. پس داده ها را از تاریخ 01/01/2009 تا تاریخ 01/01/2010 جدا می نماییم سپس پنجره ی اول داده ی شماره یک تا پنج خواهد بود و برای پنجره ی دوم داده ی دو تا شش درنظر گرفته می شود. و به همین ترتیب پنجره های بعدی را خواهیم داشت.

با توضیح فوق بدیهی است ویژگی های آب و هوایی چهار روز اول را چون هنوز تشکیل یک پنجره نمی دهنند نمی توان پیش بینی نمود. همچنین در بخش دیگری از سوال دمای روزی را برحسب روز قبل نیز خواسته است. پس در کل دو پنجره زمانی خواهیم داشت یکی برای پنج روز دیگری برای یک روز

توضیح کد: پس از آن که داده های آموزش و آزمون را از یکدیگر تفکیک کردیم ستون های غیر عددی در اینجا تاریخ و ماه را نیز حذف می کنیم. سپس یک تابع `generate_sequence_windows` تعریف می کنیم تا به ازای طول پنجره ی زمانی و تعداد `step` دیتا فریم مورد نظر را به چندین پنجره تقسیم بندی نماید. در نهایت با دستور `dffrancetrain.iloc[4 :].copy()` مقادیر نهایی یا هدف را می یابیم.

کد اول: ایجاد توالی های لغزشی

```
def generate_sequence_windows(dataframe, window_length, step=1):
    import numpy as np
    data = dataframe.select_dtypes(include=[np.number]).to_numpy()
    total_rows, num_features = data.shape
    windows = np.lib.stride_tricks.sliding_window_view(data, (window_length, num_features))
    nruter = windows.reshape(-1, window_length, num_features)[::step]
```

این تابع داده را به توالی های لغزشی با طول مشخص (`window_length`) و گام (`step`) تقسیم می کند. خروجی یک آرایه ۳۰۰۰ شامل توالی های زمانی است.

کد دوم: آماده سازی توالی های آموزشی و آزمایشی



```
train_sequences = generate_sequence_windows(df_france_train, window_length=5, step=1)
test_sequences = generate_sequence_windows(df_france_test, window_length=5, step=1)
train_targets = df_france_train.iloc[4:].copy()
test_targets = df_france_test.iloc[4:].copy()
```

توالی‌ها برای داده‌های آموزشی و آزمایشی با طول ۵ ایجاد شده و مقادیر هدف از ردیفهای بعدی انتخاب می‌شوند تا هم راستا با توالی‌ها باشند.

کد سوم: نمایش ابعاد داده‌ها

```
tnirp(train_targets.shape)
tnirp(train_sequences.shape)
tnirp(test_sequences.shape)
tnirp(test_targets.shape)
```

ابعاد داده‌های ورودی و هدف برای اطمینان از تطابق بررسی می‌شود. این ابعاد باید به‌طور صحیح با تعداد توالی‌ها و ویژگی‌ها مطابقت داشته باشد.

خروجی به شکل زیر خواهد بود:

```
# Display shapes for verification
print(train_targets.shape)
print(train_sequences.shape)
print(test_sequences.shape)
print(test_targets.shape)

(3284, 25)
(3284, 5, 24)
(361, 5, 24)
(361, 25)
```

شکل ۴: سایز داده‌های آزمون و آموزش

مشاهده می‌شود تاریخ 01/01/2009 جزو داده‌های آموزش در نظر گرفته نشده است تعداد سطر‌ها یا نمونه‌های دادگان آموزش و آزمون به شکل فوق خواهد بود.

۲.۱ آموزش مدل

۱.۲.۱

برای داده‌های چند منبعی و پراکنده روش یادگیری ماشین گروهی یا همکارانه روشی مناسب است زیرا می‌تواند از چندین مدل به ازای ایستگاه‌های مختلف آب و هوایی با همکاری هم و بطور همزمان برای نتیجه ای بهتر بهره ببرد. این مدل‌ها بطور مستقل از هم اطلاعات خود را با سایر مدل‌ها به اشتراک می‌گذارند تا بهترین پیش‌بین بدست آید. این روش همچنین باعث بهبود امنیت داده‌ها می‌شود زیرا داده‌ها در یک مکان واحد ذخیره نمی‌گردند. استفاده از منبع توزیع شده هزیته‌ها را نیز کاهش می‌دهد زیرا ضرروت نیاز به زیرساخت برای پردازش داده بصورت متمرکز از بین می‌رود. در مقاله برای هر یک از ایستگاه‌های آب و هوایی یک مدل و برای پیش‌بینی جهانی نیز مدل‌هایی دیگر داریم که با یکدیگر همکاری می‌کنند تا ویژگی‌های آب و هوایی را بهتر پیش‌بینی کنند. دو نمونه برای مدل‌های ایستگاه آب و هوایی *LSTM*, *MLP* می‌باشد. همچنین دو نمونه از مدل‌های پیش‌بینی جهانی استفاده شده در این مقاله مدل‌های *ERAS*, *GFS* می‌باشد. به عبارت دیگر نتایج پیش‌بینی‌های مختلف با یکدیگر ترکیب شده و پیش‌بینی نهایی بدست می‌آید. این ترکیب می‌تواند بصورت میانگین گیری وزنی، انتخاب بهترین پیش‌بینی، استفاده از یک مدل یادگیری ثانویه باشد. این نوع از روش‌های ماشین لرنینگ برای پیش‌بینی‌های بلادرنگ مناسب است زیرا عملکردی سریع تر و دقیق تر دارد. همچنین امنیت داده‌ها را تضمین می‌نمایند.



۳.۱

توضیح کد:

توضیح کد پیاده‌سازی مدل رگرسیون

در این بخش سه مرحله‌ی اصلی از کد پیاده‌سازی شده برای پیش‌بینی سری‌های زمانی توضیح داده می‌شود:

۱. آماده‌سازی داده‌ها:

■ برای حذف ستون‌های غیر عددی (مثل `Timestamp`) از دستور `select_dtypes(include=[np.number])` استفاده شده است.

■ داده‌های سری زمانی به دو روش آماده شده‌اند:

(آ) در روش `One-step` از تابع `make_single_step_data(df)` برای ایجاد ویژگی‌ها (X) و اهداف (y) استفاده شده که در آن $df[t] = y$ و $df[t-1] = X$ است.

(ب) در روش `Windowed` از تابع `generate_sequence_windows(df, window_length=5)` استفاده شده که پنجره‌هایی از داده‌های قبلی با طول مشخص ایجاد می‌کند. سپس داده‌های ۳ بعدی با استفاده از دستور `reshape(..., 1)` به شکل ۲ بعدی تبدیل می‌شوند.

■ هدف‌ها (`labels`) برای مدل پنجره‌ای با استفاده از `iloc[window_length:-1:]` از داده‌های اصلی جدا شده‌اند.

۲. آموزش مدل‌ها:

■ دو نمونه از کلاس `CustomRegressor` با پارامترهای `stop_threshold=0.001`, `lr=0.01` و `epochs=500` مقداردهی اولیه شده‌اند.

■ آموزش مدل با استفاده از دستور `train(X, y)` انجام شده که طی آن وزن‌ها با گرادیان نزولی بهروزرسانی شده و مقدار خطای میانگین مربعات (MSE) در هر مرحله ثبت می‌شود.

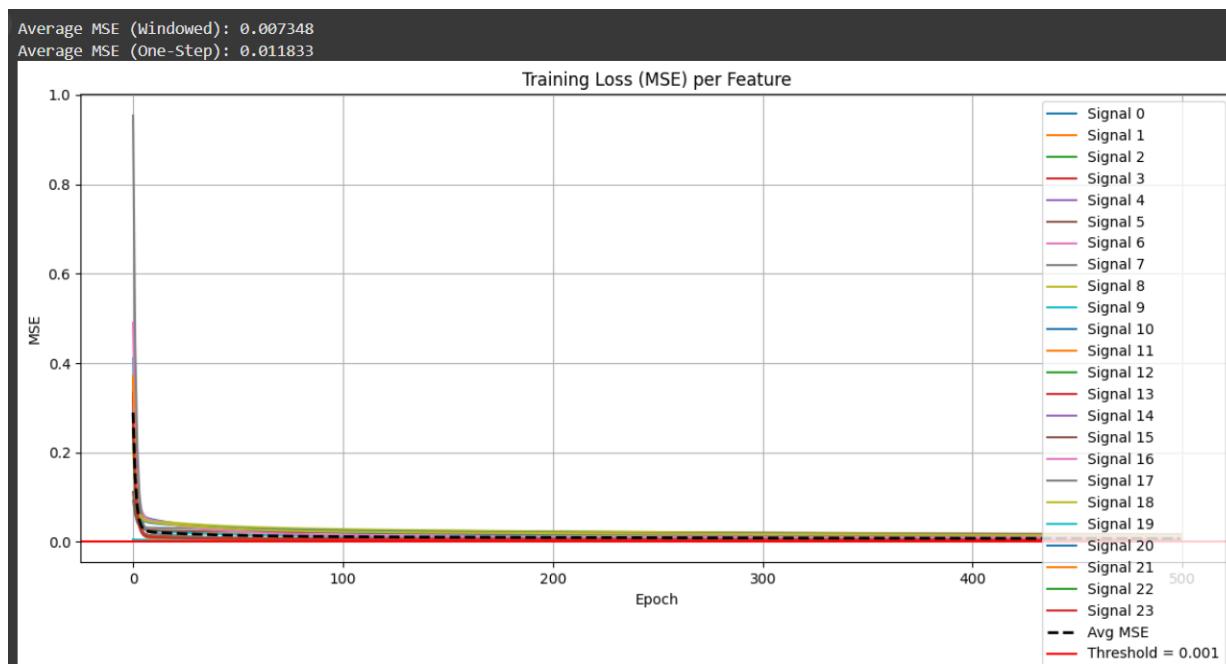
۳. پیش‌بینی، ارزیابی و مصوّرسازی:

■ پیش‌بینی با استفاده از دستور `predict(X)` برای داده‌های تست انجام شده است.

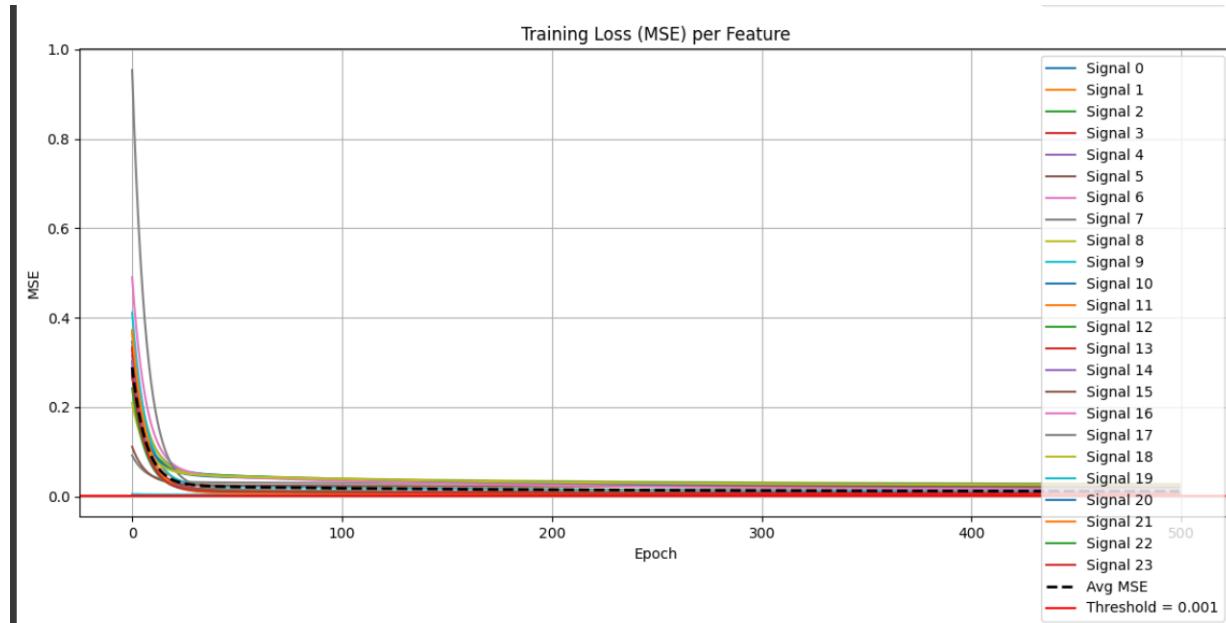
■ مقدار MSE برای هر ویژگی با استفاده از رابطه `pred.mean(axis=0) - np.mean(pred, ** true)` محاسبه شده و میانگین کلی نیز گرفته شده است.

■ برای نمایش روند کاهش خطای آموزش استفاده شده که نمودار خطی از خطای آموزش در طول دوران آموزش را نمایش می‌دهد.

خروجی برای آستانه 0.001 و تعداد تکرار 500 و نرخ یادگیری



شکل ۵: مجموع مربعات خطأ برای پنجره ۵ تایی



شکل ۶: مجموع مربعات خطأ برای onestep

مطابق شکل فوق میانگین مجموع مربعات خطأ برای حالت پنجره که از پنج داده با همپوشانی چهارتایی استفاده می کند بطور قابل توجهی کمتر از حالت یک مرحله ای (بر حسب ویژگی های آب و هوا) دیروز ویژگی های آبو هوا را امروز را تعیین می کند) است.

۱.۳.۱

چهار رگرسیون انتخابی *Linear, Lasso, Ridge, Elasticnet* می باشد.



ابتدا به تشریح رگرسیون خطی می پردازیم. در رگرسیون خطی که یکی از ابتدایی ترین الگوریتم های ماشین لرنینگ می باشد، رابطه‌ی بین ویژگی‌ها و متغیرهای هدف با یک خط مدل می شود. با در نظر گرفتن متغیرهای هدف y و متغیرهای مستقل x مدل خطی زیر را خواهیم داشت:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \epsilon \quad (1)$$

در هنگام آموزش مدل ضرایب را از روش حداقل مربعات بصورت زیر تخمین می زنند و تابع هزینه رگرسیون خطی برای تخمین ضرایب به صورت زیر است:

$$\hat{\beta} = (X^T X)^{-1} X^T y \quad (2)$$

از مزایای این روش سادگی، قابلیت تقسیر و اجرای سریع است. از معایب آن می توان به حساسیت بالا به داده‌های پرت، وقوع چندخطی در نقص داده و عدم استفاده از این روش برای رابطه‌ی غیر خطی متغیرهای اشاره کرد.

سپس به تشریح رگرسیون لسو پرداخته می شود. بطور کلی از این نوع رگرسیون برای جلوگیری از بیش برازش در مدل‌های با ابعاد بالا که تعداد ویژگی‌ها از نمونه‌ها بیشتر است استفاده می شود. در این نوع رگرسیون به تابع هزینه اندازه مطلقی به عنوان جریمه برای کاهش اندازه ضرایب اضافه می شود و ضریب لاندا اندازه‌ی آن را تعیین می کند. در فرمول زیر بتا ضریب هر ویژگی و m تعداد نمونه‌ها و n تعداد ویژگی‌ها است:

$$\min_{\beta} \left(\frac{1}{2} \sum_{i=1}^m (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^n |\beta_j| \right) \quad (3)$$

با کمی دقت در می یابیم اگر لاندا برابر صفر شود این رگرسیون لسو در واقع همان رگرسیون خطی می شود. با افزایش لاندا ضرایب بتا به صفر میل می کند. از مزایای این روش کاهش ابعاد و سادگی، جلوگیری از بیش برازش و تفسیر آسان را می توان نام برد. و معایب آن احتمال ایجاد زیر برازش است. حال رگرسیون Ridge مشروح می گردد. این روش برای حل مشکل چند هم خطی با ضرایب پایدار تر است. تابع هزینه آن به شکل زیر می باشد:

$$\min_{\beta} \left(\frac{1}{2} \sum_{i=1}^m (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^n \beta_j^2 \right) \quad (4)$$

از مزایای این روش می توان به بهبود پیش‌بینی و پایداری در حضور چند هم خطی اشاره نمود. و معایب آن این است که به علت صفر نشدن ضرایب مدل نهایی ممکن است پیچیده گردد.

و آخرین رگرسیون رگرسیون الاستیک نت بررسی می شود. این رگرسیون در واقع ترکیبی از رگرسیون لاسو و ریدج است به گونه‌ای که بطور همزمان از مزایای هر دو روش فوق می توان بهره برد. فرمول آن نیز به شکل زیر است:

$$\text{Minimize}_{\beta} \sum_{i=1}^n \left(Y_i - \hat{Y}_i \right)^2 + \lambda_1 \sum_{j=1}^p |\beta_j| + \lambda_2 \sum_{j=1}^p \beta_j^2$$

کمترین مجموع مربعات خطا برای چهار رگرسیون فوق به شکل زیر خواهد بود:



```
Model Evaluation (Windowed Data):
Linear Regression: MSE = 0.00000000
Ridge: MSE = 0.0001845
Lasso: MSE = 0.03424516
ElasticNet: MSE = 0.03122186
```

شکل ۷: مربعات خطأ برای چهار رگرسیون

مشاهده میشود کمترین خطأ برای رگرسیون خطی می باشد.

Extra ۴.۱

ملاحظه بفرمایید در اخر این سوال برای پنجره های زمانی یک روز و پنج روز برای دمای شهر تورس با استفاده از اطلاعات فقط شهر تورس و اطلاعات حاصل از سه شهر بصورت یادگیری همکارانه نیز مدل آموزش دیده شد و خطای های MSE , $MAPE$ بدست آمد و میزان کارایی این روش ها با یکدیگر مقایسه گردید همچنین نمودار های مقایسه ای دمای پیش بینی شده در این حالا با مقدار واقعی دما در آن روز نیز ترسیم گردید.



۲ تشخیص عیب یاتاقان غلتشی بر مبنای دست هبندی های سلسله مراتبی

۱.۲ دادگان

۱.۱.۲

سازوکار داده برداری این مجموعه داده را شرح دهد. برای داده برداری از چه سنسورهایی استفاده شده است؟

مجموعه داده *MaFaulDa*

مجموعه داده *MaFaulDa* در شرایط کنترل شده آزمایشگاهی برای مطالعه خرابی در ماشینهای دوار جمعآوری شده است. هدف، شبیه‌سازی خرابی‌های واقعی در حالی بود که سیگنال‌های ارتعاشی با دقت بالا ثبت می‌شدند.

پیکربندی جمعآوری داده‌ها:

- از یک میز تست شامل اجزای دوار مانند موتور، شفت، بیبرینگ، چرخ‌دنده و وزنهای اضافه شده برای شبیه‌سازی خطأ استفاده شد.
- ماشین‌ها در شرایط مختلف سرعت و بار آزمایش شدند.
- سیگنال‌ها در هر دو حالت سالم و معیوب با شدت‌های متفاوت خطأ ثبت شدند.

حسگرهای استفاده شده:

۱. ستاپ سنج سه‌محوره (ستون‌های ۲ تا ۷): ارتعاشات را در سه جهت محوری، شعاعی و مماسی اندازه‌گیری می‌کند.
۲. تاکومتر یا انکودر (ستون اول): برای اندازه‌گیری سرعت چرخش (*RPM*) یا فرکانس.
۳. میکروفون (ستون هشتم)

این حسگرهای امکان تحلیل الگوهای ارتعاشی را فراهم می‌کنند که به تشخیص ناهمانگی، عدم تعادل، سایش بیبرینگ و موارد دیگر کمک می‌کند.

توضیحات تکمیلی ساز و کار و ستاپ آزمایش:

این ستاپ دارای اجزای اصلی زیر است:

موتور الکتریکی.

روتور.

یاتاقان‌ها: دو یاتاقان اصلی در بخش جلویی و زیرین شفت قرار دارند.

حسگرهای حسگرها نقش مهمی در جمعآوری داده‌های عملکردی سیستم دارند. این حسگرها شامل ۶ عدد ستاپ سنج *IMISensors601A01* (برای اندازه‌گیری ارتعاش در جهت‌های شعاعی، محوری و مماسی بر روی یاتاقان جلویی و پشتی)، یک عدد میکروفون *Shure1SM* (برای ثبت صدا)، و یک عدد سرعت‌سنج *MonarchInstrumentMT – 190* (برای اندازه‌گیری دقیق نرخ چرخش شفت) هستند.

دادگان آزمایشی در این ستاپ به این شکل جمعآوری می‌شوند:

یک وزنه به بخش‌های مختلف سیستم اضافه می‌شود و با این عمل، ستاپ فعال می‌شود. این وزنهای اضافه شده، هر کدام نشان‌دهنده یک نوع خرابی در سیستم هستند. این خرابی‌ها موجب ایجاد ارتعاش، تغییر دما یا تغییر در انتقال سرعت در سیستم می‌شوند. تغییرات حاصله توسط سنسورها ثبت شده و برای تحلیل‌های بعدی ذخیره می‌شوند.



توضیح کلاس‌های معیوب

در این مجموعه از داده، سه نوع خطأ مورد بررسی قرار گرفته‌اند که به شرح زیر هستند:

ناهمراستایی در دو جهت افقی و عمودی، با محدوده‌ای بین ۵۰ تا ۲ میلی‌متر.

عدم تعادل از طریق اضافه کردن وزن‌هایی در بازه ۶ تا ۳۵ گرم به روتور.

یاتاقان‌ها در سه دسته عیب قفس، عیب بیرونی و عیب توپ، با اعمال وزن‌هایی بین ۰ تا ۳۵ گرم.

توجه: ایجاد خطأ در یاتاقان‌ها شامل بررسی عیوب در هر دو یاتاقان جلو و یاتاقان پشت می‌شود. در نتیجه مجموعه دادگان شامل ده کلاس با عنوان‌های زیر می‌باشد.

normal, imbalance, misalignment horizontal, misalignment vertical, under-hang(ball, cage, race outer), over-hang(ball, cage, race outer)

با توجه به اینکه تجهیز مورد مطالعه دارای حرکت دورانی است، سرعت دورانی نقش مهمی در جمع آوری داده‌ها ایفا می‌کند. به همین دلیل، سرعت‌های مختلفی در محدوده ۷۳۷ تا ۳۸۸۶ دور بر دقیقه با گام‌های ۶۰ تایی مورد بررسی قرار گرفته‌اند. این رویکرد به حدود ۵۰ آزمایش برای هر نوع خطأ منجر شده است.

پس از ایجاد انواع خطاهای، فرآیند جمع آوری داده‌ها با نزخ نمونه‌برداری ۵۰ کیلوهertz آغاز می‌شود. این فرآیند منجر به ثبت ۲۵۰،۰۰۰ داده برای هر عیوب در هر دور می‌شود.

با توجه به مطالب بیان شده، هر خطأ شامل ۵۰ آزمایش است و هر آزمایش به صورت یک جدول داده‌ای با ۲۵۰،۰۰۰ ردیف و ۸ ستون سازمان‌دهی شده است. در این جدول، ستون اول مربوط به سرعت چرخش است. ستون‌های دوم تا چهارم شامل داده‌های ارتعاشی در جهت‌های عمودی، شعاعی و محوری برای یاتاقان جلو هستند. ستون‌های پنجم تا هفتم نیز داده‌های ارتعاشی یاتاقان پشتی را نشان می‌دهند و ستون هشتم مربوط به داده‌های صوتی ثبت شده توسط میکروفون است.



۲.۱.۲

کلاس های مختلف عیب را که در این دادگان جمع آوری شده است را معرفی کرده و توضیح دهید. چه مقدار داده برای هر بخش ثبت شده است مجموعه داده شامل ۶ دسته‌ی اصلی از خطاهای است که هر کدام دارای سطوح مختلفی از شدت یا نوع خطا هستند.

دسته‌های خطا و ساختار پوشه‌ها:

:horizontal-misalignment .۱

سطح: 2.0mm ,1.5mm ,1.0mm ,0.5mm

:vertical-misalignment .۲

سطح: 1.78mm ,1.40mm ,1.27mm ,0.63mm ,0.51mm

:imbalance .۳

سطح: 30g ,25g ,20g ,15g ,10g

:normal .۴

فایل‌های سی‌اس‌وی به صورت مستقیم در پوشه قرار دارند (مثال: 13.1072.csv ,12.288.csv و ...)

:overhang .۵

انواع خط: outer_race ,cage_fault ,ball_fault

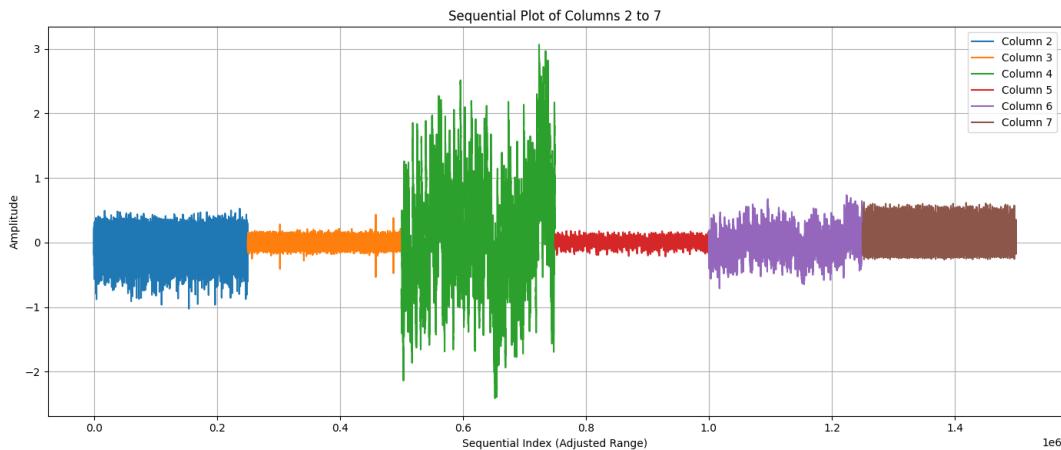
:underhang .۶

انواع خط: outer_race ,cage_fault ,ball_fault

هر زیرپوشه شامل چندین فایل CSV است که هر کدام نمایانگر یک نمونه سیگنال ارتعاشی هستند.

۳.۱.۲

تمامی کلاس های عیبی که در پایان نامه بررسی شده اند را در نظر گرفته و به تعداد دلخواه، فایل داده از این پیوند دانلود کنید. دقت کنید که لازم نیست تمامی فایل های داده برای هر کلاس را استفاده کنید، صرفا یک یا دو فایل داده برای این مبنی پروژه کفایت می کند. سیگنال های موجود در یک فایل را همانند شکل 2 – ۳ پایان نامه نمایش داده و هر بخش را با رنگ مجزا و لیبل مشخص کنید ما تمامی داده را لود کرده و از هر کلاس یدونه فایل خطاب انتخاب کرده ایم، در واقع دومین فایل هر لیست انتخاب شده است. همچنین فایل "mafaulda/overhang/cage-fault/6g/14.5408.csv" را به صورت رندوم نمایش دادیم:



شکل ۸: رسم بزرگی داده های یک سیگنال در مجاورت هم

۲.۲ پیش پردازش و استخراج ویژگی

۱.۲.۲

پیش پردازش داده یکی از اساسی ترین بخش های ایجاد یک مدل تشخیص عیب است. در این مرحله فعالیت هایی همانند حذف نویز یا نرمال سازی روی داده انجام می شود. با مطالعه بخش ۲ – ۳ پایان نامه، مراحل مختلف پیش پردازش داده را توضیح دهید. به نظر شما مرحله استخراج ویژگی از کدام بخش شروع شده است؟ مراحل استخراج ویژگی را کتاب گذاشته و تمامی پیش پردازش های انجام شده را روی داده خود اعمال کنید.

خلاصه پیش پردازش داده ها برای تشخیص خطأ

فرآیند پیش پردازش به صورت مستقیم بر روی سیگنال های خام ارتعاشی از مجموعه داده MaFaulDa انجام شد. عملیات پاک سازی و حذف نویز انجام نشد و تمرکز به طور مستقیم روی استخراج ساختاریافته ویژگی ها قرار گرفت.

مراحل انجام شده:

۱. نرمال سازی سیگنال:

برای هر سیگنال، ستون های ۲، ۳، ۴ و ۷ (اندازه گیری محورهای مختلف) انتخاب شدند.

$$x_{\text{normalized}} = \frac{x - \text{mean}(x)}{\text{var}(x)}$$

۲. بخش بندی سیگنال با پنجره های لغزان:

سیگنال ها به پنجره های با طول ثابت $n = 5$ تقسیم شدند.

۳. استخراج ویژگی (برای هر پنجره و هر محور انتخاب شده):



مقدار RMS (ریشه میانگین مربع)

دامنه اوج

ضریب کرست ($crest factor$)

واریانس

انرژی کل از تبدیل فوریه (FFT)

فرکانس غالب

بیشترین فرکانس بالای آستانه مشخص شده

چولگی

۴. نمونه برداری از کلاس‌ها:

از هر کلاس فقط یک سیگنال نماینده استفاده شد. این سیگنال‌ها شامل ۱۰ کلاس هدف زیر بودند:

```
normal_13.1072 .imbalance_15g .horizontal_1mm  
underhang_cage .underhang_ball .overhang_cage .overhang_ball  
vertical_0.63mm
```

۵. ذخیره‌سازی:

خروجی‌های پیش‌پردازش شده در دیکشنری با نام `data_dict[class_name]` ذخیره شدند.

هر مقدار، یک ماتریس با ابعاد `(num_windows, 40)` است.

تقسیم‌بندی آموزش و تست

داده‌های هر کلاس (بعد از استخراج ویژگی‌ها و ذخیره در X) به صورت زیر تقسیم شدند:

۸۰ درصد برای آموزش

۲۰ درصد برای آزمون

```
✓ Split horizontal-misalignment → train: (40000, 40), test: (10000, 40)  
✓ Split imbalance → train: (40000, 40), test: (10000, 40)  
✓ Split normal → train: (40000, 40), test: (10000, 40)  
✓ Split overhang_ball_fault → train: (40000, 40), test: (10000, 40)  
✓ Split overhang_cage_fault → train: (40000, 40), test: (10000, 40)  
✓ Split overhang_outer_race → train: (40000, 40), test: (10000, 40)  
✓ Split underhang_ball_fault → train: (40000, 40), test: (10000, 40)  
✓ Split underhang_cage_fault → train: (40000, 40), test: (10000, 40)  
✓ Split underhang_outer_race → train: (40000, 40), test: (10000, 40)  
✓ Split vertical-misalignment → train: (40000, 40), test: (10000, 40)  
↳ Original shape: (400000, 40)  
↳ Balanced shape: (400000, 40)
```

شکل ۹: نتایج تفکیک داده‌های تست و آزمون برای آموزش مدل پی از پیش‌پردازش



۲.۲.۲

در پایان نامه مذکور ابتدا ستون های اول و هشتم که مربوط به سرعت روتور و صدای ثبت شده توسط میکروفون هستند حذف شده‌اند زیرا در این پایان نامه هدف آن است که ارتباط بین ارتعاشات سیستم با نوع خرایی مشخص شود و در مرحله بعد ستون ۵ بخاطر اهمیت نه چندان حذف شده ما نیز پیروی میکنیم. سپس پنجره‌ای مناسب انتخاب شده و با کمک داده‌های موجود در آن پنجره، ویژگی داده‌ها مشخص می‌شود این ویژگی‌ها در دو حوزه زمان و فرکانس می‌باشند. بعد از مشخص شدن ویژگی‌ها آن‌ها استاندارد سازی می‌شوند تا الگوریتم بهینه سازی بهتر عمل کند و سریعتر همگرا شود برای اینکار میانگین داده‌ها صفر و انحراف معیار آن‌ها یک می‌شود بعد از استاندارد سازی نوبت این می‌شود که مشخص شود کدام ویژگی‌ها اهمیت بالایی دارند و ترتیب اولیت ویژگی‌ها چیست که برای اینکار از *LightGBM* کمک گرفته می‌شود و به کمک *SI* ویژگی‌های بهینه انتخاب می‌گردند. این تقسیم‌بندی بر اساس ترتیب زمانی پنجره‌ها انجام گرفت. پس از حذف ستون‌های اضافی، نوبت به استخراج ویژگی‌ها از ستون‌های باقی‌مانده می‌رسد. از آنجا که سیگنال ارتعاشی یک سیگنال سری زمانی است، در این دسته از سیگنال‌ها هم ویژگی‌های زمانی و هم ویژگی‌های فرکانسی مطرح می‌شوند. برای هر دو دسته ویژگی‌ها روش‌های در نظر گرفته شده است. البته باید توجه داشت که برای استخراج ویژگی‌ها باید یک پنجره زمانی انتخاب کرد و داده‌ها را به بخش‌هایی به اندازه این پنجره تقسیم نمود، سپس ویژگی‌ها از هر بخش استخراج شوند. در این پروژه، اندازه پنجره برابر با ۵ در نظر گرفته شده است.

بخش امتیازی رتبه بندی ویژگی‌ها با *LightGBM*, *SI*معرفی الگوریتم *LightGBM*

الگوریتم *Light Gradient Boosting Machine* (LightGBM) یک الگوریتم یادگیری تقویتی مبتنی بر درخت تصمیم است که توسط شرکت Microsoft توسعه داده شده است. این الگوریتم برای مسائل دسته‌بندی و رگرسیون استفاده می‌شود و به خاطر سرعت بالا و دقت مناسب، به ویژه در مجموعه‌داده‌های بزرگ و با ابعاد بالا محبوبیت یافته است.

ویژگی‌های اصلی *LightGBM*:

استفاده از الگوریتم Gradient Boosting برای ساخت مدل‌های قوی از ترکیب مدل‌های ضعیف رشد درخت به صورت **leaf-wise** به جای **level-wise** که باعث افزایش دقت می‌شود پشتیبانی از دسته‌بندی چندکلاسه، داده‌های نامتوازن، و قابلیت تنظیم وزن کلاس‌ها مصرف حافظه پایین و مقیاس‌پذیری بالا

کاربردها

دسته‌بندی خطاهای صنعتی

تحلیل ریسک

تشخیص تقلب

رتبه‌بندی در موتورهای جستجو

معرفی شاخص سیلوئت (*SilhouetteIndex*)

Silhouette Index یک معیار برای ارزیابی کیفیت خوشبندی در الگوریتم‌های بدون ناظر مانند K-Means است. این شاخص نشان می‌دهد که هر داده چقدر به خوشی خود تعلق دارد و تا چه اندازه از خوشهای دیگر فاصله دارد.

فرمول محاسبه:

فرض کنیم برای هر نقطه i :



(a) میانگین فاصله‌ی نقطه i با سایر نقاط در همان خوشهاش باشد.

(b) کمترین میانگین فاصله‌ی i با نقاط سایر خوشها باشد.

آنگاه مقدار Silhouette Index برای آن نقطه برابر است با:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

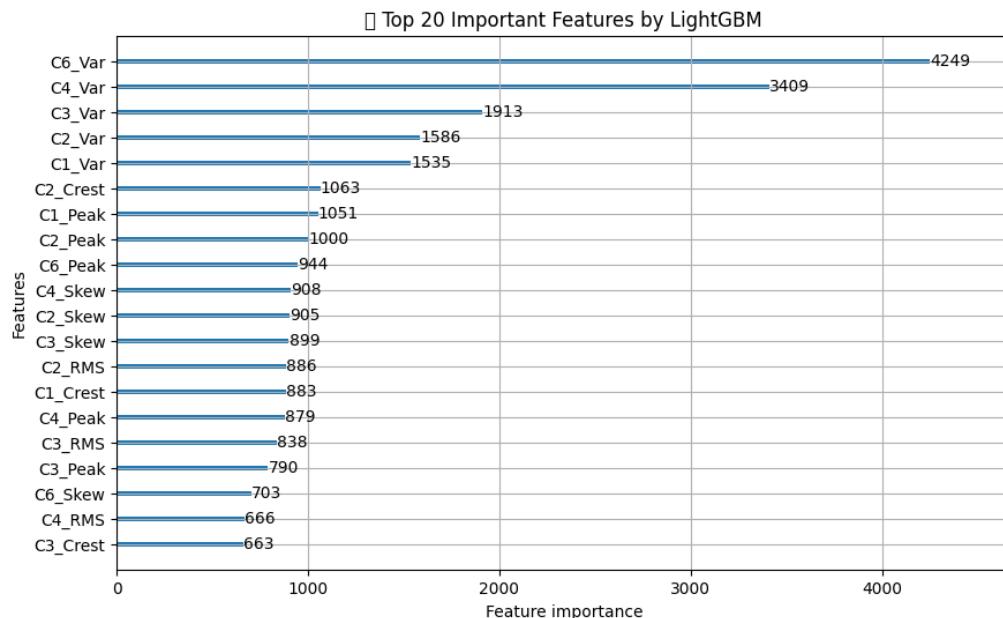
مقدار شاخص:

اگر $s(i) > 1$ باشد خوشبندی خوب است.

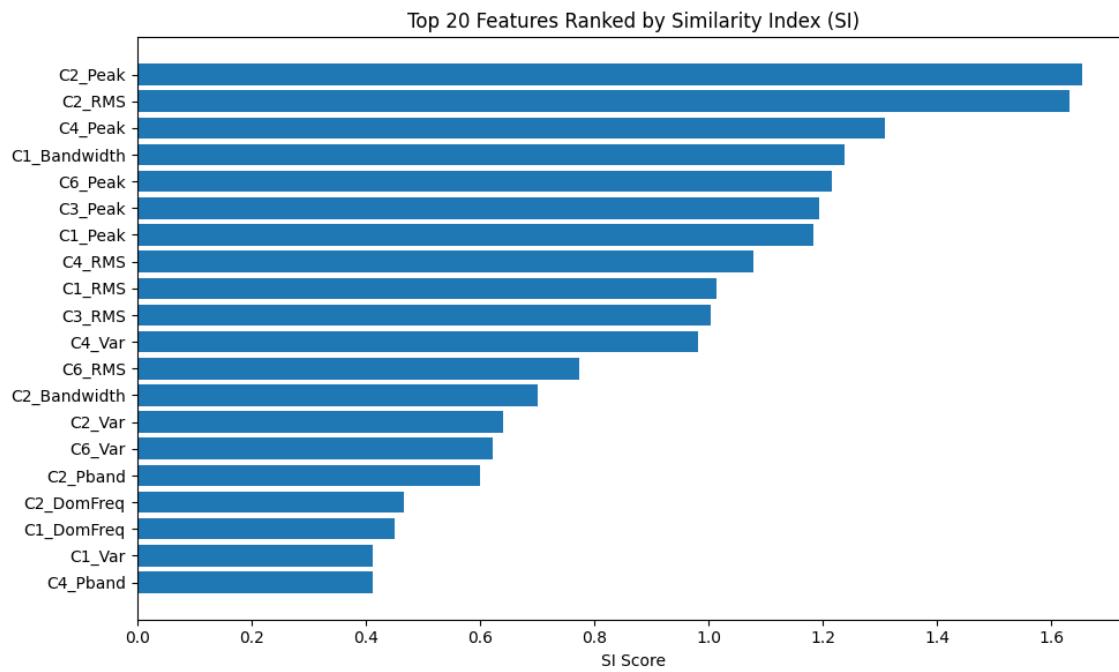
اگر $0 < s(i) < 1$ باشد نقطه در مرز بین دو خوش است.

اگر $s(i) < 0$ منفی باشد خوشبندی اشتباه انجام شده است.

نتایج به صورت زیر هستند:



شکل ۲۰: ویژگی برتر رتبه بندی شده با LightGBM

شکل ۱۱: ۲۰ ویژگی برتر رتبه بندی شده با SI



۳.۲ آموزش مدل

۱.۳.۲

مدل تشخیص سلسله‌مراتبی (چندسطحی)

یکی از ایده‌های کلیدی در این پژوهه، طراحی یک مدل تشخیص سلسله‌مراتبی (چندمرحله‌ای) بود. این رویکرد مسئله را به مراحل تصمیم‌گیری کوچک‌تر و قابل‌کنترل تقسیم می‌کند که باعث افزایش تفسیرپذیری و انعطاف‌پذیری در فرآیند دسته‌بندی می‌شود.

مدل سلسله‌مراتبی چیست؟

مدل سلسله‌مراتبی شامل چندین مرحله تصمیم‌گیری است که هر مرحله بر روی یک بخش خاص از مسئله تمرکز می‌کند. در این پژوهه، ساختار مدل به شکل زیر در نظر گرفته شده است:

مرحله ۱: تشخیص سالم یا معیوب بودن سیگنال.

مرحله ۲: در صورت وجود خطأ، شناسایی دسته کلی خطأ (مانند عدم تعادل یا ناهمانگی).

مرحله ۳: دسته‌بندی دقیق‌تر خطأ (مثالاً `cage_fault` یا `underhang_ball_fault`).

نحوه استفاده از مدل سلسله‌مراتبی در این تحقیق

در این تحقیق:

یک ساختار مرحله‌به‌مرحله برای دسته‌بندی بر اساس رفتار واقعی ماشین‌ها پیشنهاد شد.

در هر مرحله، ویژگی‌های مرتبط و مهم انتخاب شدند تا مدل تنها از مؤثرترین داده‌ها استفاده کند.

از روش‌های انتخاب ویژگی `LightGBM` و `SI` در مراحل مناسب استفاده شد تا دقت نهایی بهبود یابد.

مزایای این رویکرد

دقت بالاتر به‌واسطه ساده‌تر شدن تصمیم‌گیری در هر مرحله.

تفسیرپذیری بهتر و امکان تحلیل راحت‌تر رفتار مدل.

ساختار مازوچار که هر مرحله می‌تواند به صورت مستقل بازآموزی شود.

قابلیت انطباق بهتر با داده‌های نامتوازن، از طریق تفکیک کلاس‌های اقلیت در مراحل ابتدایی.

آیا این رویکرد مؤثر بود؟

بله نتایج نشان داد که استفاده از ساختار سلسله‌مراتبی:

باعث بهبود عملکرد کلی مدل در دسته‌بندی شد.

میزان اشتباه در تفکیک کلاس‌های مشابه را کاهش داد.

تعیین‌پذیری بهتری را در مواجهه با داده‌های دیده‌نشده فراهم کرد.

در نتیجه، استفاده از این رویکرد سلسله‌مراتبی تأیید و پیشنهاد می‌شود.

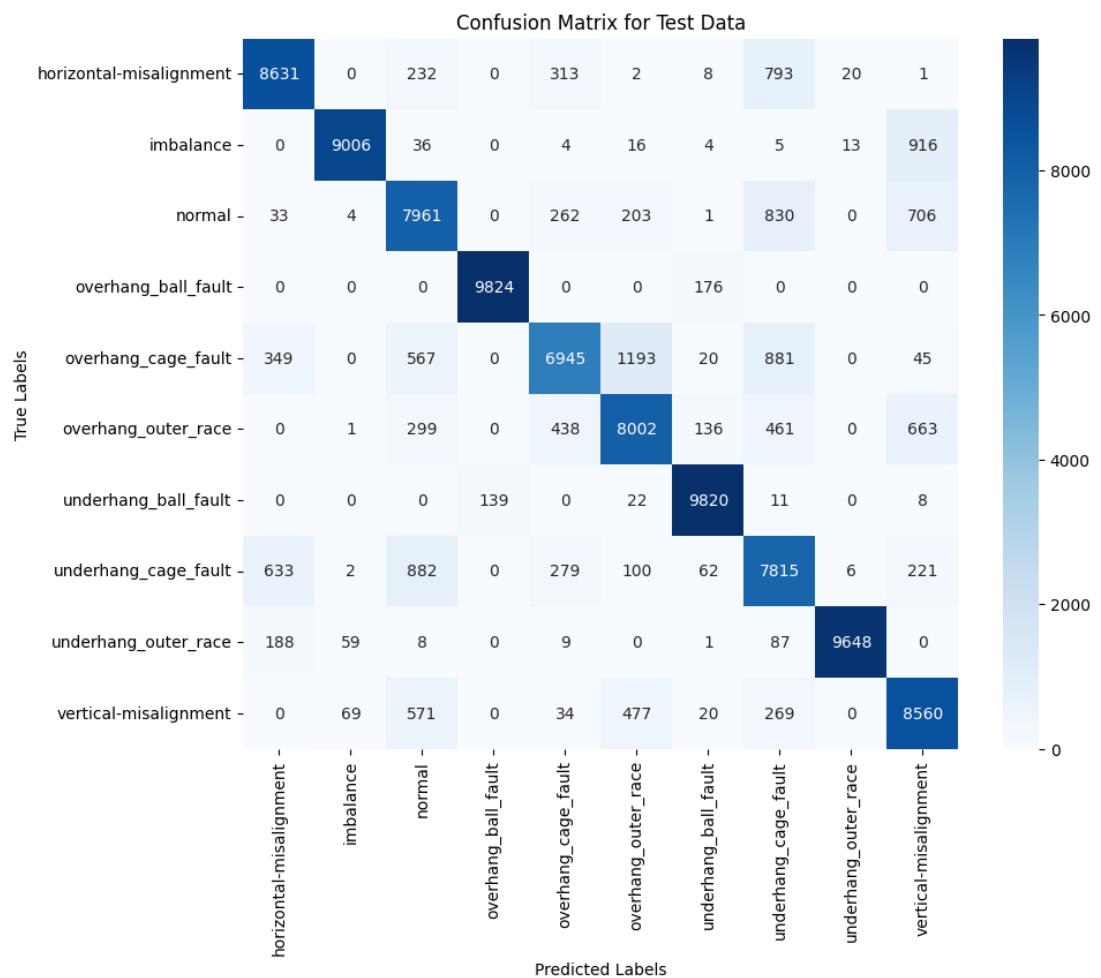
۲.۳.۲

نتایج رویکرد اول: بدون استفاده از ساختار سلسله مراتبی

ماتریس های آشتفتگی رو برای داده های آزمون و تست رسم میکنیم:

Confusion Matrix for Train Data											
True Labels	horizontal-misalignment	34482	3	848	0	1306	18	50	3199	79	15
	imbalance	0	35736	136	0	27	68	9	17	32	3975
	normal	140	13	31688	0	1010	810	6	3397	0	2936
	overhang_ball_fault	0	0	0	39128	0	0	872	0	0	0
	overhang_cage_fault	1378	2	2094	0	27805	4861	77	3615	0	168
	overhang_outer_race	10	11	1175	0	1600	32080	539	1894	0	2691
	underhang_ball_fault	0	1	0	436	0	113	39357	54	0	39
	underhang_cage_fault	2514	3	3437	0	1227	468	233	31085	32	1001
	underhang_outer_race	680	243	28	0	37	0	11	388	38613	0
	vertical-misalignment	0	278	2163	0	185	2039	72	1088	0	34175

شکل ۱۲: ماتریس آشتفتگی داده های آزمون



شکل ۱۳: ماتریس آشفتگی داده های تست

ضمما امتیاز مدل برای هر دو در حدود ۰.۸۶ بوده است.



نتایج رویکرد دوم: با استفاده از ساختار سلسله مراتبی

در مرحله اول، طبقه‌بند اول وظیفه دارد تا داده‌ها را بین ۵ گروه اصلی تقسیم کند که شامل اند از:

۱. ناهمراستایی (*misalignment*)
۲. عدم تعادل (*imbalance*)
۳. حالت نرمال (*normal*)
۴. اورهنج (*overhang*)
۵. آندرهنج (*underhang*)

این مرحله به درستی موفق شد داده‌ها را در سطح کلی دسته‌بندی کند و دقت خوبی در شناسایی تفاوت بین حالت سالم و انواع کلی خطأ از خود نشان داد.

در مرحله دوم، داده‌ای که در گروه «ناهماهنگی» قرار گرفته بودند، بین دو کلاس ناهماهنگی افقی و ناهماهنگی عمودی تقسیم شدند. به دلیل تفاوت‌های مشخص در ساختار سیگنال این دو کلاس، مدل توانست با دقت مناسبی این دو نوع خطأ را تشخیص دهد.

برای داده‌های مربوط به «اورهنج» و «آندرهنج»، یک مرحله‌ی طبقه‌بندی دیگر انجام شد تا نوع خطای بیرونی تعیین گردد. این خطاهای شامل خرابی قفسه‌ای، خرابی ساقمه‌ای و خرابی سطح بیرونی بودند. در این مراحل، به دلیل شباهت بالای الگوهای ارتعاشی میان برخی کلاس‌ها، دقت طبقه‌بندی کمی کاهش یافت، اما همچنان عملکرد قابل قبولی ارائه شد.

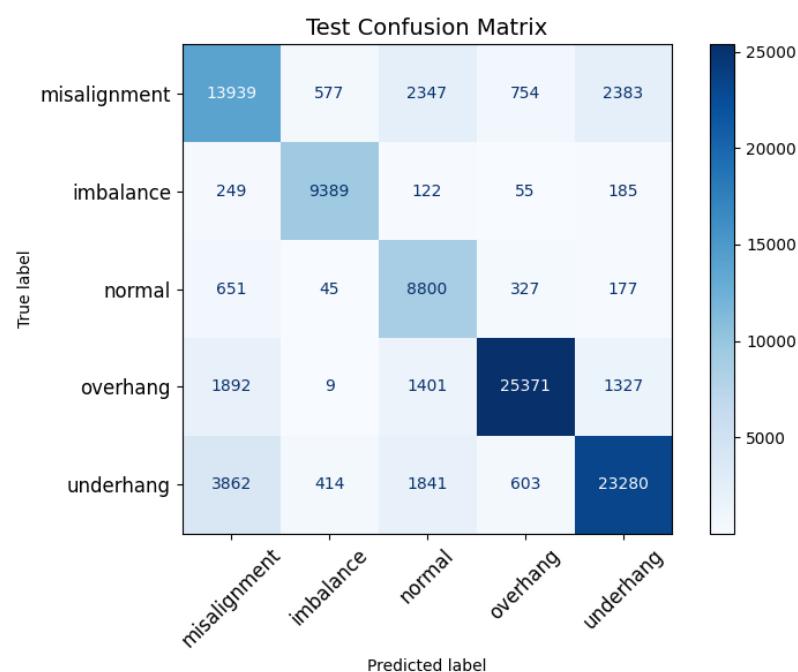
در مجموع، ساختار سلسله‌مراتبی باعث شد که مدل در مواجهه با تصمیمات ساده‌تر، عملکرد بهتری داشته باشد و طبقه‌بندها با تمرکز بر زیورفضاهای خاصی از مسئله، دقت بیشتری را فراهم کنند. همچنین این ساختار موجب کاهش پیچیدگی مسئله‌ی اصلی و بهبود قابلیت تعمیم مدل شد.

نتایج ماتریس های آشفتگی را در زیر مشاهده میکنیم:

ماتریس های آشفتگی آزمون و تست طبقه بند اول



شکل ۱۴: ماتریس آشفتگی داده های آزمون طبقه بند اول



شکل ۱۵: ماتریس آشفتگی داده های تست طبقه بند اول



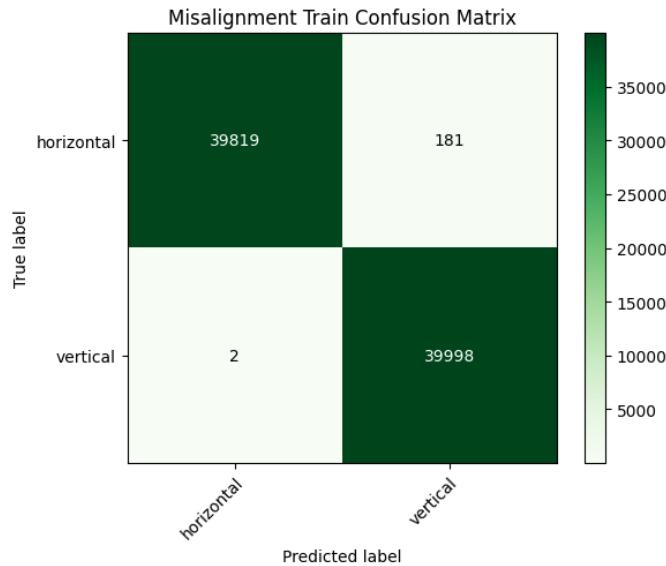
```

Level-1 Accuracy (Test): 0.8078
Level-1 Accuracy (Train, Balanced): 0.8259
Level-1 Train Shape: (60000, 40)
Level-1 Label Map: {0: 'misalignment', 1: 'imbalance', 2: 'normal', 3: 'overhang', 4: 'underhang'}

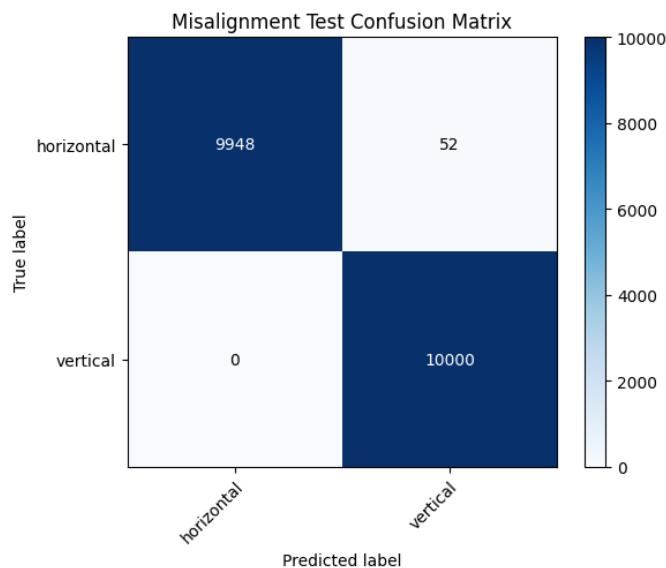
```

شکل ۱۶: امتیاز مدل طبقه بند اول

ماتریس های آشنتگی آزمون و تست طبقه بند دوم



شکل ۱۷: ماتریس آشنتگی داده های آزمون طبقه بند دوم

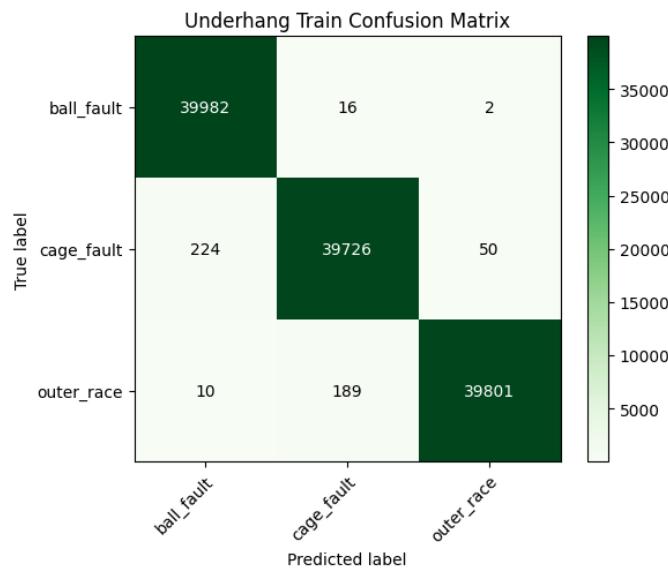


شکل ۱۸: ماتریس آشنتگی داده های تست طبقه بند دوم

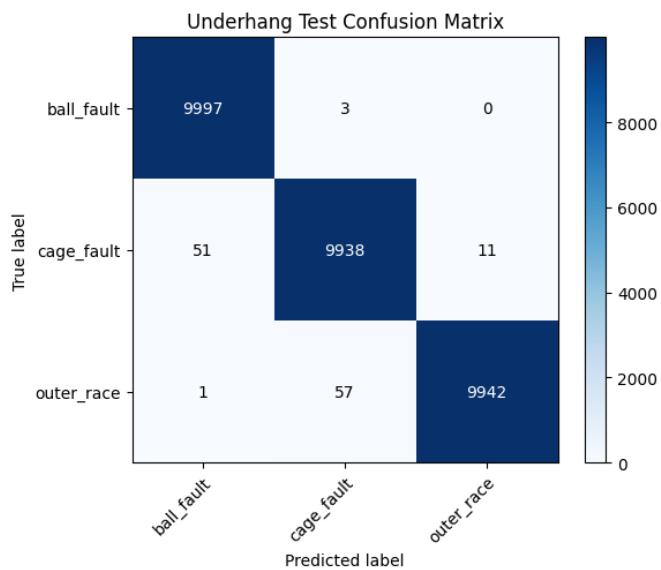
در ضمن امتیاز مدل ۰.۹۹۷۴ شده است.



ماتریس های آشتفتگی آزمون و تست طبقه بند سوم



شکل ۱۹: ماتریس آشتفتگی داده های آزمون طبقه بند سوم

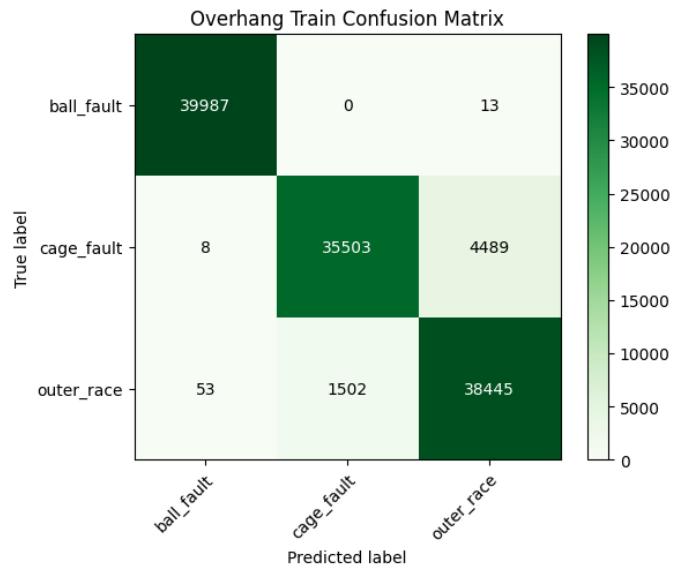


شکل ۲۰: ماتریس آشتفتگی داده های تست طبقه بند سوم

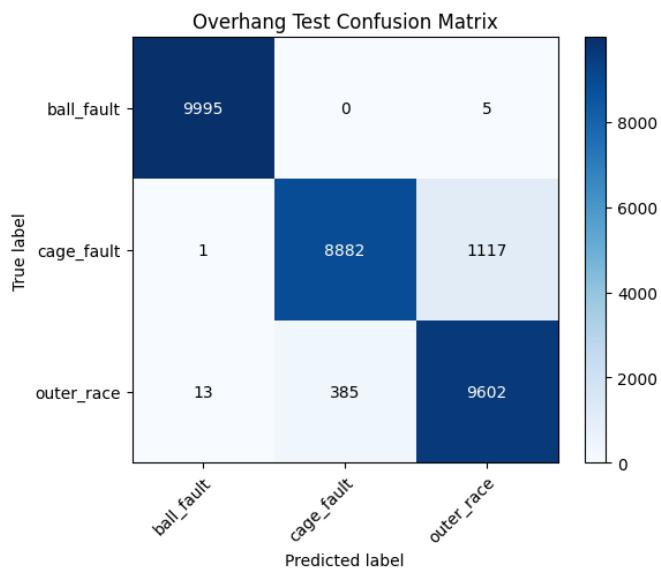
در ضمن امتیاز مدل ۰.۹۹۵۹ شده است.



ماتریس های آشفتگی آزمون و تست طبقه بند چهارم



شکل ۲۱: ماتریس آشفتگی داده های آزمون طبقه بند چهارم



شکل ۲۲: ماتریس آشفتگی داده های تست طبقه بند چهارم

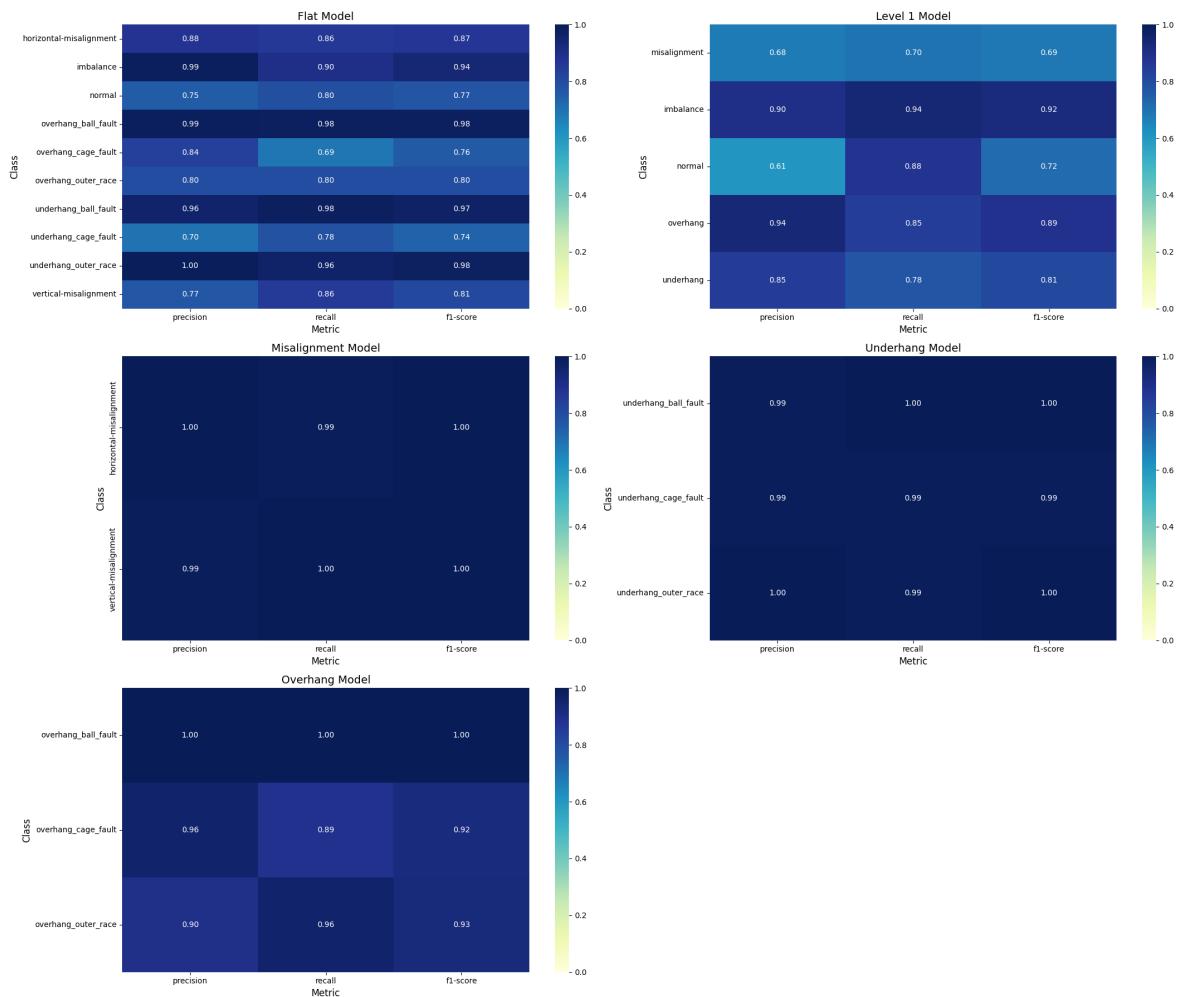
در ضمن امتیاز مدل ۰.۹۴۹۳ شده است.



۳.۳.۲

مقایسه و بررسی گزارش های طبقه بندی هر مدل

سه ویژگی هر مدل: Precision , Recall , F1-Score را به صورت گرافیکی به تصویر کشیدیم



شکل ۲۳: نمودار گزارش طبقه بندی هر مدل

همانطور که مشاهده میشود ساختار سلسله مراتبی بسیار دقیقتر عمل کرده و همینطور کم هزینه تر است، این امر ارجح بودن این ساختار را نمایان میکند. گزارش مفصل تر هر مدل در خروجی مربوطه در کوب نشان داده شده است.



۴.۳.۲

با در نظر گرفتن تمامی جنبه های رویکردهای پیش بردہ شده در بخش قبل، شما از کدام روش استفاده می کنید؟ دلیل خود را توضیح دهید.

انتخاب نهایی و تحلیل مقایسه ای رویکردها

با بررسی دقیق دو رویکرد ارائه شده در این پژوهه یعنی طبقه بندی مستقیم و ساختار سلسله مراتبی، می توان نتیجه گرفت که استفاده از مدل سلسله مراتبی (Hierarchical Classification) گزینه مناسب تری برای تشخیص عیب در یاتاقان های چرخشی است. زیرا:

رویکرد سلسله مراتبی با شکستن مسئله به مراحل کوچک تر، امکان مدیریت بهتر تصمیم گیری را فراهم می کند.

دقت مدل در سطوح مختلف افزایش یافته است، زیرا هر مرحله به یک زیرمجموعه خاص از داده ها متوجه شده و ویژگی های متناسب انتخاب شده اند.

مدل های هر سطح به صورت مستقل قابل آموزش و بهبود هستند، بنابراین در صورت تغییر یا به روزرسانی در یک دسته خاص، کل مدل نیازی به بازآموزی ندارد.

عملکرد مدل بر روی داده های آزمون بهتر از مدل مستقیم بوده است، و ماتریس های درهم ریختگی (Confusion Matrix) نشان می دهند که تشخیص میان دسته های مشابه بهتر انجام گرفته است.

با وجود افزایش پیچیدگی ساختار، ساختار سلسله مراتبی تفسیر پذیری مدل را نیز افزایش داده است.

در نتیجه، استفاده از ساختار سلسله مراتبی به عنوان روش نهایی پیشنهاد می شود، چرا که علاوه بر عملکرد دقیق تر، قابلیت نگهداری، گسترش پذیری و تفسیر بهتر را نیز فراهم می کند.



۴.۲ مخصوص

برنامه طراحی شده با ساختار سلسله مراتبی به گونه ایست که ابتدا یکبار کل داده تصادفی را طبقه بندی میکند و گزارش میدهد که چه تعداد پنجره را به کدامین کلاس طبقه بندی کرده است.

سپس کد مجزا برای طبقه بندی کردن یک پنجره خاص از داده تصادفی نوشته شده که برنامه ابتدا از طبقه بند اول استفاده میکند و نتیجه را گزارش میدهد. اگر نتیجه حاصله زیرطبقه ای نداشته باشد کار به اتمام میرسد، اما اگر زیر طبقه داشته باشد بسته به نوع آن مدل مربوط به آن سطح را که پیشتر آموزش دادیم فراخوانی میکند و نتیجه زیرطبقه را اعلام میکند.

```

▶ # Step 1: Choose the test class
selected_class_path = "underhang_ball_fault" # Replace with any test_data key

# Step 2: Get all samples from that class (each is a windowed feature vector)
all_samples = test_data[selected_class_path] # This is already a NumPy array

# Step 3: Classify all windows using the hierarchical classifier
predicted_labels = []

for sample in all_samples:
    label = predict_fault(sample, {
        'level1': level1_model,
        'misalignment': mis_model,
        'underhang': underhang_model,
        'overhang': overhang_model
    })
    predicted_labels.append(label)

# Step 4: Report results
print(f"Classified {len(all_samples)} windows from class '{selected_class_path}'")

# Optional: Count how many times each class was predicted
from collections import Counter
print("Prediction distribution:")
print(Counter(predicted_labels))

→ Classified 10000 windows from class 'underhang_ball_fault'
Prediction distribution:
Counter({'underhang_ball_fault': 9812, 'overhang_ball_fault': 105, 'overhang_outer_race': 75, 'imbalance': 6, 'underhang_cage_fault': 2})

```

شکل ۴.۲۴: تعداد پنجره های دارای طبقه بندی صحیح و غلط در کل داده تصادفی



کد محصول برای طبقه بندی پنجره تصادفی و خروجی آن را در زیر مشاهده خواهید کرد:

```

▶ # Step 1: Pick a sample window from your test set
selected_class_path = "underhang_outer_race" # or any valid key from your test_data
sample_window_index = 1120 # index of the test window to classify

sample = test_data[selected_class_path][sample_window_index].reshape(1, -1)

# Step 2: Predict Level 1 (coarse category)
lvl1_pred_idx = level1_model.predict(sample)[0]
lvl1_pred_class = label_map_lvl1[lvl1_pred_idx]

print(f"🟡 Level 1 predicted class: {lvl1_pred_class}")

# Step 3: Level 2 - handle misalignment (horizontal vs vertical)
if lvl1_pred_class == "misalignment":
    lvl2_pred_idx = mis_model.predict(sample)[0]
    lvl2_pred_class = label_map_misalign[lvl2_pred_idx]
    print(f"🔵 Level 2 predicted class (misalignment): {lvl2_pred_class}")

# Step 4: Level 3 - handle underhang faults
elif lvl1_pred_class == "underhang":
    lvl3_pred_idx = underhang_model.predict(sample)[0]
    lvl3_pred_class = label_map_underhang[lvl3_pred_idx]
    print(f"🔴 Level 2 predicted class (underhang): {lvl3_pred_class}")

# Step 5: Level 4 - if underhang class is ball_fault or cage_fault
if lvl3_pred_class in ["underhang_ball_fault", "underhang_cage_fault"]:
    lvl4_pred_idx = underhang_fine_model.predict(sample)[0]
    lvl4_pred_class = label_map_underhang_fine[lvl4_pred_idx]
    print(f"🟠 Level 3 predicted class (underhang fine): {lvl4_pred_class}")

# Step 6: Level 3 - handle overhang faults
elif lvl1_pred_class == "overhang":
    lvl3_pred_idx = overhang_model.predict(sample)[0]
    lvl3_pred_class = label_map_overhang[lvl3_pred_idx]
    print(f"🟠 Level 2 predicted class (overhang): {lvl3_pred_class}")

# Step 7: Level 4 - if overhang class is ball_fault or cage_fault
if lvl3_pred_class in ["overhang_ball_fault", "overhang_cage_fault"]:
    lvl4_pred_idx = overhang_fine_model.predict(sample)[0]
    lvl4_pred_class = label_map_overhang_fine[lvl4_pred_idx]
    print(f"🟠 Level 3 predicted class (overhang fine): {lvl4_pred_class}")

# If no deeper classification needed
else:
    print("✅ No further classification needed at deeper levels.")

➡️ 🟡 Level 1 predicted class: underhang
➡️ 🔵 Level 2 predicted class (underhang): underhang_outer_race

```

شکل ۲۵: محصول و خروجی آن برای داده تصادفی