دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده مهندسی برق

مینی پروژه ۱ یادگیری ماشین

نگارش

رضا لاری

مهدی جوکار

استاد درس

دکتر علیاری

فروردین ۱۴۰۴

لینک گوگل کولب:

سوال ۱:

https://colab.research.google.com/drive/1W6VvF1wd85mkX3a6OGjJ1xoZ72k2qjX4?authuser=3#scrollTo=z1E7Rk3ACZzS

سوال ۲:

https://colab.research.google.com/drive/12mctztMyGZiFtXbmrRqGI9is0IQ8VxqH?authuser=1#scrollTo=wupaYFvpFCff

سوال ۱) پیش بینی آب و هوا مبتنی بر یادگیری ماشین

(۱.۱.۱

داده‌ها از چندین مکان مختلف برای پیش‌بینی وضعیت آب‌وهوا در یک مکان خاص استفاده شده‌اند. در این پژوهش، پنج الگوریتم یادگیری ماشین به کار گرفته شده و آزمایش‌هایی در چهار منطقه مختلف در موریس برای پیش‌بینی پارامترهای آب‌وهوایی مانند دما، سرعت و جهت باد، فشار، رطوبت و میزان ابری بودن انجام شده است. داده‌های هواشناسی با استفاده از API سرویس OpenWeather از طریق یک دستگاه لبه‌ای (Edge Device) موبایل و یک دستگاه دسکتاپ جمع‌آوری شده‌اند. این داده‌ها به‌صورت فایل JSON در پایگاه داده **IBM Cloudant** و یک پایگاه داده محلی **MySQL** ذخیره شده‌اند. تحلیل داده‌ها هم در یک سرور محلی که داده‌های ورودی از دستگاه لبه‌ای را دریافت می‌کند و هم از طریق یک سرولت مستقر در پلتفرم ابری **IBM** انجام شده است.

(۲.۱.۱

این کار در section 1.1.2 انجام شده است. شهرهای فرانسه موجود در دادگان عبارتند از:

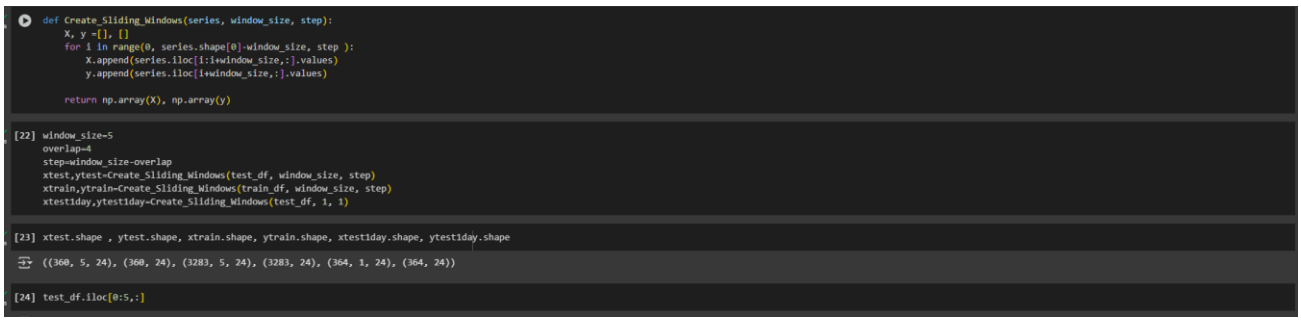
Montelimar,Perpignan,Tours

```
[ ] search_columns = [col for col in weather_prediction_dataset.columns \
      if any(keyword in col for keyword in ["MONTELIMAR", "PERPIGNAN", "TOURS"])]
    # extracts only the columns whose headers contain Montelimar, Perpignan, or Tours

    search_columns

['MONTELIMAR_wind_speed',
 'MONTELIMAR_humidity',
 'MONTELIMAR_pressure',
 'MONTELIMAR_global_radiation',
 'MONTELIMAR_precipitation',
 'MONTELIMAR_temp_mean',
 'MONTELIMAR_temp_min',
 'MONTELIMAR_temp_max',
 'PERPIGNAN_wind_speed',
 'PERPIGNAN_humidity',
 'PERPIGNAN_pressure',
 'PERPIGNAN_global_radiation',
 'PERPIGNAN_precipitation',
 'PERPIGNAN_temp_mean',
 'PERPIGNAN_temp_min',
 'PERPIGNAN_temp_max',
 'TOURS_wind_speed',
 'TOURS_humidity',
 'TOURS_pressure',
 'TOURS_global_radiation',
 'TOURS_precipitation',
 'TOURS_temp_mean',
 'TOURS_temp_min',
 'TOURS_temp_max']

[ ] new_df=weather_prediction_dataset[search_columns]
```

(۳.۱.۱

این داده شامل ۳۶۵۴*۲۴ نمونه می‌باشد. که از ۱ ژانویه ۲۰۰۰ تا ۱ ژانویه ۲۰۱۰ را شامل می‌شود. از جمله پیش پردازش‌های لازم، یافتن Null ها و اصلاح دیتاست میباشد. همچنین ستون اندیس این دیتاست به نوع Datetime تبدیل شده تا کار با آن راحت تر باشد. در نهایت نیز از آنجایی که ۱ روز از ۲۰۱۰ موجود میباشد ولی در صورت سوال تا سال ۲۰۰۹ نیاز است، آن روز حذف گردیده است.

(۴.۱.۱



```
def Create_Sliding_Windows(series, window_size, step):
    X, y =[], []
    for i in range(0, series.shape[0]-window_size, step ):
        X.append(series.iloc[i:i+window_size,:].values)
        y.append(series.iloc[i+window_size,:].values)

    return np.array(X), np.array(y)
```

```
[22] window_size=5
     overlap=4
     step=window_size-overlap
     xtest,ytest=Create_Sliding_Windows(test_df, window_size, step)
     xtrain,ytrain=Create_Sliding_Windows(train_df, window_size, step)
     xtest1day,ytest1day=Create_Sliding_Windows(test_df, 1, 1)
```

```
[23] xtest.shape , ytest.shape, xtrain.shape, ytrain.shape, xtest1day.shape, ytest1day.shape
⇒ ((360, 5, 24), (360, 24), (3283, 5, 24), (3283, 24), (364, 1, 24), (364, 24))
```

```
[24] test_df.iloc[0:5,:]
```

تابع Create_Sliding_Windows به منظور ساخت پنجره‌های زمانی متحرک (Sliding Windows) از یک مجموعه‌داده سری‌زمانی طراحی شده است. این روش یکی از تکنیک‌های رایج در یادگیری ماشین برای مدل‌سازی داده‌های وابسته به زمان است، مانند پیش‌بینی دما، قیمت یا سایر ویژگی‌هایی که در گذر زمان تغییر می‌کنند. این تابع سه ورودی اصلی دارد series :که دیتافریم اولیه حاوی داده‌های سری‌زمانی است، window_size که اندازه پنجره (تعداد ردیف‌هایی که در هر بار استفاده می‌شود) را مشخص می‌کند، و step که مشخص می‌کند هر بار پنجره به چه میزان به جلو حرکت کند (مثلاً با مقدار ۱، هر پنجره یک روز جلو می‌رود و با روز قبل هم‌پوشانی دارد).

در ابتدای تابع، دو لیست خالی به نام‌های X و y تعریف می‌شود. لیست X قرار است مجموعه پنجره‌های ورودی (توالی داده‌ها برای آموزش مدل) را نگه‌داری کند، در حالی که y شامل مقادیر هدف یا برچسب‌ها (که معمولاً داده‌ی بلافاصله پس از هر پنجره هستند) خواهد بود. این ساختار به ما اجازه می‌دهد که برای هر پنجره از داده‌ها، مقدار آینده‌ای که می‌خواهیم پیش‌بینی کنیم را نیز ذخیره کنیم.

در ادامه، یک حلقه for اجرا می‌شود که از ابتدای داده‌ها تا جایی که هنوز امکان تشکیل یک پنجره کامل وجود داشته باشد، حرکت می‌کند. در هر مرحله از این حلقه، یک پنجره با اندازه مشخص

(window_size)از ردیف‌های دیتافریم استخراج شده و با استفاده از values. به آرایه‌ی عددی تبدیل می‌شود. این پنجره به لیست Xافزوده می‌شود. سپس، مقدار مربوط به ردیف بعد از پنجره (یعنی آنچه باید پیش‌بینی شود) نیز استخراج و به لیست y اضافه می‌شود.

در پایان، لیست‌های X و y به آرایه‌های NumPy تبدیل شده و به عنوان خروجی تابع بازگردانده می‌شوند.

۲.۱)

آموزش‌ها برای شهر MONTELIMAR انجام می‌شود

۱.۲.۱)

یادگیری ماشین مشارکتی یکی از رویکردهای پیشرفته در علم داده و هوش مصنوعی است که هدف آن به‌کارگیری داده‌های پراکنده و توزیع‌شده برای آموزش یک مدل یادگیری ماشین مشترک می‌باشد. در این روش، به‌جای تمرکز صرف بر داده‌های یک منبع یا مکان خاص، داده‌هایی از منابع مختلف – که ممکن است از نظر جغرافیایی، سازمانی یا فنی از هم جدا باشند – گردآوری و استفاده می‌شوند. این نوع یادگیری معمولاً در شرایطی استفاده می‌شود که تجمیع مستقیم داده‌ها ممکن نیست یا منجر به چالش‌هایی مانند نقض حریم خصوصی، حجم بالای داده یا هزینه‌های انتقال می‌شود. مدل یادگیری مشارکتی تلاش می‌کند تا از الگوهای موجود در داده‌های پراکنده بهره بگیرد و دقت و قابلیت تعمیم مدل را بهبود بخشد.

در این مقاله، یادگیری ماشین مشارکتی به‌عنوان فرآیندی تعریف شده است که در آن از داده‌های مربوط به چند مکان مختلف برای پیش‌بینی وضعیت آب‌وهوای مورد انتظار در یک منطقه‌ی خاص استفاده می‌شود. این تعریف بر اهمیت استفاده از داده‌های چندمنطقه‌ای برای به‌دست آوردن پیش‌بینی دقیق‌تر تأکید دارد، چرا که وضعیت آب‌وهوای یک منطقه می‌تواند به‌شدت تحت تأثیر شرایط مناطق اطراف قرار گیرد. بر همین اساس، مدل یادگیری مشارکتی با بهره‌گیری از اطلاعات دمایی، رطوبت، باد و سایر

ویژگی‌های آب‌وهوایی از ایستگاه‌های مختلف، می‌تواند درک جامع‌تری از الگوهای جوی به‌دست آورد و پیش‌بینی‌های دقیق‌تری انجام دهد.

۳.۱) در ادامه کد برای رگرسیون چند جمله‌ای توضیح داده می‌شود ولی در فایل کد ارسال شده، ابتدا مدل‌های خطی پیاده سازی شده‌اند و سپس کد رگرسیون چندجمله‌ای نشان داده شده است.

مدل رگرسیون چند جمله‌ای ارائه شده چندین بخش کلیدی را شامل می‌شود: آماده‌سازی داده‌ها با استفاده از پنجره‌های متحرک(Sliding Windows) ، انتخاب ویژگی‌ها براساس همبستگی با متغیر هدف، ایجاد ویژگی‌های چندجمله‌ای (Polynomial Features) و در نهایت آموزش مدل رگرسیون چندجمله‌ای از صفر با استفاده از گرادیان کاهشی. در ادامه، هر بخش به تفصیل توضیح داده شده است.

۱. آماده‌سازی داده‌ها و ساخت پنجره‌های متحرک

ابتدا تابع Create_Sliding_Windows تعریف شده است. این تابع از یک دیتافریم ورودی (که حاوی داده‌های سری زمانی است) استفاده می‌کند و به کمک دو پارامتر window_size (اندازه پنجره؛ مثلاً تعداد روزها) و step (گام حرکت پنجره) داده‌ها را به پنجره‌های همپوشانی تقسیم می‌کند. در هر تکرار حلقه، یک بخش پیوسته (به اندازه‌ی window_size) از داده‌ها استخراج شده و به لیست X اضافه می‌شود؛ در حالی که سطر بعدی به عنوان برچسب (هدف) برای پیش‌بینی، به لیست y اضافه می‌شود. در پایان، این دو لیست به آرایه‌های NumPy تبدیل می‌شوند تا برای پردازش‌های بعدی آماده باشند. این روش اجازه می‌دهد تا مدل بتواند از تاریخچه‌ای کوتاه (مثلاً ۵ روز) استفاده کرده و روز بعد را پیش‌بینی کند.

۲. صاف‌سازی (Flattening) پنجره‌ها

تابع flatten_windows داده‌های تولیدشده توسط تابع قبلی (با ابعاد num_windows×window_size×num_features) را به یک آرایه دوبعدی تبدیل می‌کند. در این تابع، با استفاده از متد reshape، هر پنجره به یک بردار یک‌بعدی درآورده می‌شود. به عبارت دیگر، اگر

هر پنجره شامل ۵ روز با ۲۴ ویژگی باشد، پس از صاف‌سازی هر نمونه به برداری با ۵×۲۴=۱۲۰ ویژگی تبدیل می‌شود.

```python
def flatten_windows(X):
    """
    Flatten each window in X (shape: num_windows x window_size x num_features)
    into a 2D array (num_windows x (window_size*num_features)).
    """
    num_windows, window_size, num_features = X.shape
    return X.reshape(num_windows, window_size * num_features)
```

۳. انتخاب ویژگی‌ها بر اساس همبستگی

تابع select_featuresبه کمک ماتریس همبستگی که با استفاده از متد corr() به دست می‌آید، ویژگی‌هایی را انتخاب می‌کند که با متغیر هدف ("MONTELIMAR_temp_mean") دارای همبستگی مطلق بالاتر از آستانه مشخص‌شده (مثلاً ۰.۴) هستند. در نتیجه، لیستی از اسامی ستون‌ها (ویژگی‌ها) تهیه می‌شود که می‌توانند به عنوان ورودی‌های مدل استفاده شوند. انتخاب ویژگی‌ها به بهبود عملکرد مدل کمک می‌کند زیرا فقط ویژگی‌های تاثیرگذار در مدل لحاظ می‌شوند.

```python
def select_features(df, target, threshold=0.4):
    """
    Select features that have a correlation above the given threshold with the target variable.

    Args:
    df (pd.DataFrame): The input DataFrame.
    target (str): The target column name.
    threshold (float): The minimum absolute correlation required.

    Returns:
    list: Selected feature names including the target variable.
    """

    # Compute correlation matrix
    corr_matrix = df.corr()

    # Select features with absolute correlation above threshold
    selected_features = corr_matrix[target][abs(corr_matrix[target]) > threshold].index.tolist()

    return selected_features
```

```
selected=select_features(new_df, "MONTELIMAR_temp_mean")
selected
```

```
['MONTELIMAR_humidity',
 'MONTELIMAR_global_radiation',
 'MONTELIMAR_temp_mean',
 'MONTELIMAR_temp_min',
 'MONTELIMAR_temp_max',
 'PERPIGNAN_global_radiation',
 'PERPIGNAN_temp_mean',
 'PERPIGNAN_temp_min',
 'PERPIGNAN_temp_max',
 'TOURS_humidity',
 'TOURS_global_radiation',
 'TOURS_temp_mean',
 'TOURS_temp_min',
 'TOURS_temp_max']
```

```
[35] #Getting Indx Numbers
selectindex = [new_df.columns.get_loc(col) for col in selected]
selectindex
```

```
[1, 3, 5, 6, 7, 11, 13, 14, 15, 17, 19, 21, 22, 23]
```

۴. ایجاد ویژگی‌های چندجمله‌ای (Polynomial Features)

تابع create_polynomial_features وظیفه ایجاد فضای ویژگی چندجمله‌ای تا درجه مشخص (در اینجا درجه ۲) را بر عهده دارد. در این تابع ابتدا یک ستون بایاس (مقدار ثابت برابر ۱) به‌عنوان اولین ستون اضافه می‌شود. سپس، ویژگی‌های اصلی (ترم‌های خطی) به آرایه اضافه می‌شوند. در ادامه، یک حلقه تو در تو اجرا می‌شود تا ترم‌های درجه دوم که شامل تبیین‌های مربعی هر ویژگی به همراه ترم‌های مشترک (ضرب‌های دوجمله‌ای بین دو ویژگی) هستند، محاسبه و به لیست ویژگی‌ها اضافه گردد. در نتیجه، فضای ویژگی به شدت افزایش یافته و مدل خطی قادر به انطباق با روابط غیرخطی میان ویژگی‌ها خواهد بود.

```
def create_polynomial_features(X, order=2):
    """
    Create polynomial features for X up to a given order.
    Here we implement for order=1 (linear) and order=2 (quadratic).
    X is of shape (n_samples, n_features).
    Returns a new matrix of shape (n_samples, new_features) including a bias term.
    """
    n_samples, n_features = X.shape
    # Always include the bias (constant term)
    poly_features = [np.ones((n_samples, 1))]

    # Order 1: Linear terms
    poly_features.append(X)

    if order >= 2:
        # Order 2: Quadratic and interaction terms
        for i in range(n_features):
            for j in range(i, n_features):
                term = (X[:, i] * X[:, j]).reshape(-1, 1)
                poly_features.append(term)

    # For orders >2, you could extend here.
    return np.hstack(poly_features)
```

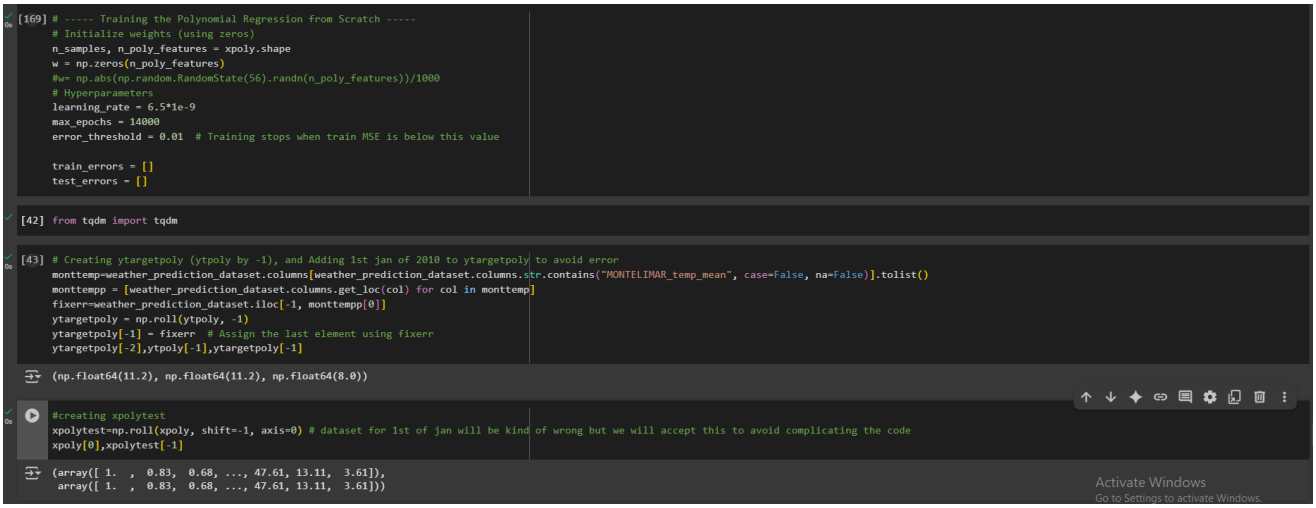
۵. آماده‌سازی داده‌های نهایی برای آموزش

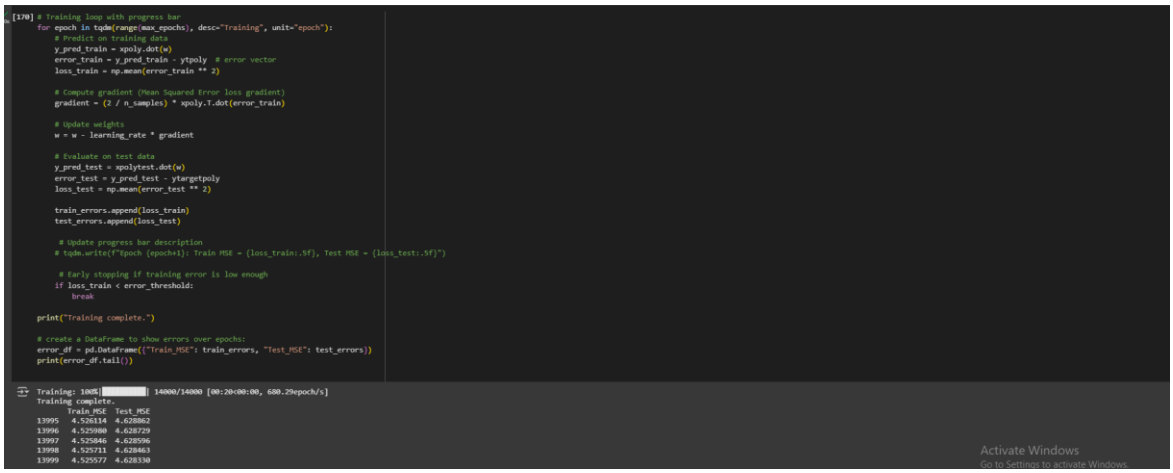
پس از ایجاد پنجره‌های متحرک، داده‌ها صاف‌سازی شده و ویژگی‌های انتخاب شده استخراج می‌شوند. سپس داده‌های صاف‌شده (که هر کدام یک نمونه از پنجره‌های متحرک هستند) با استفاده از تابع

flatten_windows به شکل دوبعدی در می‌آیند. سپس از ستون مربوط به "MONTELIMAR_temp_mean" به عنوان متغیر هدف استفاده می‌شود. در ادامه، ویژگی‌های منتخب از دیتافریم اصلی استخراج شده و اندیس‌های آن‌ها به دست می‌آید. سپس با استفاده از تابع create_polynomial_features فضای ویژگی چندجمله‌ای ایجاد می‌شود. یک عملیات شیفت نیز (با استفاده از تابع np.roll) برای هم‌ترازی درست ورودی‌ها و برچسب‌ها در نظر گرفته شده است.

۶. آموزش مدل رگرسیون چندجمله‌ای با گرادیان کاهشی

در این بخش، مدل رگرسیون چندجمله‌ای (Polynomial Regression) با استفاده از گرادیان کاهشی از ابتدا آموزش داده می‌شود. ابتدا وزن‌ها (w) به صورت آرایه‌ای از اعداد صفر به اندازه‌ی تعداد ویژگی‌های چندجمله‌ای (که پس از ایجاد ویژگی‌های چندجمله‌ای به دست آمده) مقداردهی اولیه می‌شوند. سپس پارامترهایی مانند نرخ یادگیری(learning_rate) ، حداکثر تعداد دوره‌های آموزشی (max_epochs) و آستانه خطا (error_threshold) تنظیم می‌شوند. در حلقه آموزشی که با استفاده از کتابخانه tqdm همراه با یک نوار پیشرفت اجرا می‌شود، در هر دوره پیش‌بینی‌های مدل محاسبه شده و خطای میانگین مربعات (MSE) بر روی داده‌های آموزشی و آزمون ارزیابی می‌شود. گرادیان MSE محاسبه و سپس وزن‌ها به کمک نرخ یادگیری به روز می‌شوند. آموزش زمانی متوقف می‌شود که خطای آموزشی از آستانه تعیین‌شده پایین‌تر رود. همچنین در انتهای آموزش یک DataFrame از خطاها (برای مثال، خطای دوره‌های آموزشی و آزمون) ایجاد و نمایش داده می‌شود.





۷. ارزیابی نهایی و مقایسه‌ی نتایج

در پایان آموزش، مدل نهایی بر روی ورودی‌های چندجمله‌ای نهایی عمل می‌کند و پیش‌بینی‌هایی به دست می‌دهد. سپس با استفاده از معیارهایی مانند MAPE (خطای درصد مطلق متوسط) و MAE (خطای مطلق متوسط)، عملکرد مدل ارزیابی می‌شود. همچنین یک DataFrame نهایی ایجاد می‌شود که در آن مقادیر واقعی "MONTELIMAR_temp_mean"، مقادیر پیش‌بینی شده و تفاوت آن‌ها نمایش داده می‌شود.

```
[172] from sklearn.metrics import mean_absolute_error

[173] mape = mean_absolute_percentage_error(ytargetpoly, y_pred_test)
      mae=mean_absolute_error(ytargetpoly, y_pred_test)
      mape,mae

(0.19215508920634544, 1.7216727060032013)
```

```
dfpoly = pd.DataFrame({'Actual_Large_Window': ytargetpoly, 'Prediction_Large_Window': y_pred_test})
dfpoly['Diff'] = dfpoly['Actual_Large_Window'] - dfpoly['Prediction_Large_Window']
dfpoly["MAPE"]=mean_absolute_percentage_error(dfpoly["Actual_Large_Window"],dfpoly["Prediction_Large_Window"])
dfpoly.index=test_df.index[6:].append(pd.Index([pd.to_datetime('2010-01-01')])) #1st jan 2010 was missing from test_df leading to length mismatch
dfpoly
```

	Actual_Large_Window	Prediction_Large_Window	Diff	MAPE
2009-01-07	-1.3	1.303337	-2.603337	0.192155
2009-01-08	-0.9	0.830623	-1.730623	0.192155
2009-01-09	1.2	1.605253	-0.405253	0.192155
2009-01-10	1.6	1.407137	0.192863	0.192155
2009-01-11	2.0	1.954578	0.045422	0.192155
...	...	...	...	...
2009-12-28	3.6	6.870051	-3.270051	0.192155
2009-12-29	9.4	6.484740	2.915260	0.192155
2009-12-30	10.8	8.180028	2.619972	0.192155
2009-12-31	11.2	8.564806	2.635194	0.192155
2010-01-01	8.0	1.725683	6.274317	0.192155

360 rows × 4 columns

Next steps: Generate code with dfpoly | View recommended plots | New interactive sheet

نتایج در صورت استفاده از ۱ روز قبل به صورت زیر می‌باشد:

```
Training: 100%|██████████| 14000/14000 [00:00<00:00, 15693.20epoch/s]Training complete.
       Train_MSE   Test_MSE
13995  11.163733  11.251970
13996  11.163654  11.251891
13997  11.163575  11.251811
13998  11.163496  11.251732
13999  11.163417  11.251652
```

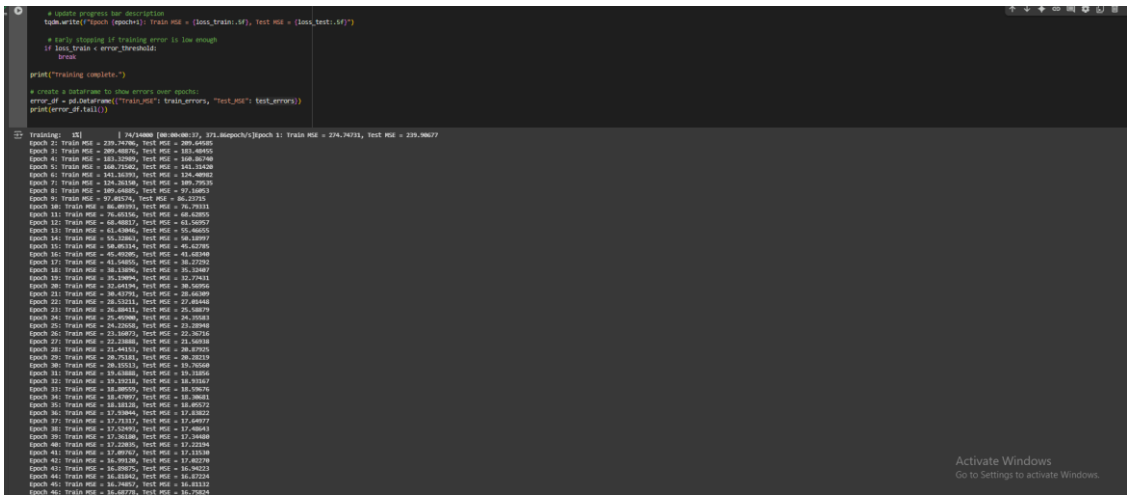
```
dfpoly = pd.DataFrame({'Actual_Large_Window': ytargetpoly, 'Prediction_Large_Window': y_pred_test})
dfpoly['Diff'] = dfpoly['Actual_Large_Window'] - dfpoly['Prediction_Large_Window']
dfpoly["MAPE"]=mean_absolute_percentage_error(dfpoly["Actual_Large_Window"],dfpoly["Prediction_Large_Window"])
dfpoly
```

	Actual_Large_Window	Prediction_Large_Window	Diff	MAPE
0	2.8	1.735735	1.064265	0.260267
1	1.8	1.179231	0.620769	0.260267
2	-0.4	1.963616	-2.363616	0.260267
3	0.2	1.217853	-1.017853	0.260267
4	-1.3	0.494812	-1.794812	0.260267
...	...	...	...	...
359	3.6	5.889826	-2.289826	0.260267
360	9.4	3.918697	5.481303	0.260267
361	10.8	6.197298	4.602702	0.260267
362	11.2	5.516250	5.683750	0.260267
363	8.0	2.328749	5.671251	0.260267

364 rows × 4 columns

همانگونه که مشاهده می‌شود، خطا در صورتی که از روزهای کمتری برای پیش‌بینی استفاده شود بیشتر است.

درخواست مربوط به نکته ۳ در شکل زیر نمایش داده شده است ولی به دلیل حجم بالا در شکل‌های قبل نمایش داده نشد:



(۱.۳.۱

در ادامه به بررسی سه مدل رگرسیون خطی موجود در کتابخانه‌ی scikit-learn می‌پردازیم.

مدل‌های انتخاب شده، رگرسیون خطی معمولی (Ordinary Least Squares - OLS)، ریدج رگرسیون (Ridge Regression) و لاسو رگرسیون (Lasso Regression) می‌باشند هر یک از این مدل‌ها از لحاظ تئوری با استفاده از اصول بهینه‌سازی برای تعیین ضرایب خطی w به کار می‌روند، اما تفاوت‌های اصلی آن‌ها در نحوه مدیریت مسئله بیش‌برازش (Overfitting) و تنظیم ضرایب با استفاده از جریمه‌های منظم‌سازی (Regularization) است.

در **رگرسیون خطی معمولی (OLS)**، تلاش بر این است تا ضرایب w و در صورت نیاز، بایاس b را به‌گونه‌ای تعیین کند که مجموع مربعات خطا (خطای بین پیش‌بینی مدل و مقادیر واقعی) کمینه شود. فرض کنید X ماتریس ویژگی‌ها با ابعاد n*p (n تعداد نمونه و p تعداد ویژگی‌ها) و y بردار مقادیر هدف باشد. فرمول مسئله به شکل زیر است:

$$J(w) = \sum_{(i=1)}^{n} (y_i - \hat{y}_i)^2 = \|y - Xw\|_2^2$$

که در آن $\hat{y}_i = x_i^T w$، جواب تحلیلی آن به صورت زیر است:

$$w = (X^T X)^{-1} X^T y.$$

ریدج رگرسیون، مدلی مبتنی بر رگرسیون خطی معمولی است که یک اصطلاح منظم‌سازی L2 به تابع هزینه اضافه می‌کند تا مشکل بیش‌برازش و ناپایداری ضرایب در مواجهه با ویژگی‌های همبسته کاهش یابد. تابع هزینه در ریدج به صورت زیر تعریف می‌شود:

$$J(w) = \|y - Xw\|_2^2 + \alpha\|w\|_2^2,$$

که در آن آلفا پارامتر منظم سازی است. با اضافه شدن جمله مربوط به آلفا، مدل برای مقادیر بزرگ w مجازات قوی تری اعمال میکند. جواب تحلیلی آن نیز به صورت زیر است:

$$w = (X^TX + \alpha I)^{-1} X^Ty.$$

لاسو رگرسیون نیز بر پایه رگرسیون خطی است، اما به جای منظم‌سازی L2 از منظم‌سازی L1 استفاده می‌کند. تابع هزینه لاسو به شکل زیر تعریف می‌شود:

$$J(w) = \|y - Xw\|_2^2 + \alpha\|w\|_1,$$

که در آن $\|w\|_1 = \sum_{(j=1)}^{p} |w_j|$ است. استفاده از منظم‌سازی L1 باعث می‌شود برخی از ضرایب دقیقاً صفر شوند و ویژگی‌هایی که اطلاعات مفیدی ندارند حذف شوند. این ویژگی باعث به وجود آمدن مدل‌های پراسپرم (Sparse) می‌شود که تفسیری ساده‌تر و کارآمدتر در مواجهه با تعداد بالای ویژگی‌ها دارند. لازم به ذکر است که برخلاف رگرسیون خطی و ریدج، در لاسو به طور عمومی جواب تحلیلی بسته وجود ندارد و باید از روش‌های بهینه‌سازی عددی مانند الگوریتم‌های کوئردیک استفاده کرد.

ابتدا از Ridge Regression استفاده میکنیم، توضیحات مربوط به دو روش دیگر مشابه میباشد:

هدف از این کد پیش‌بینی متغیر «MONTELIMAR_temp_mean» با استفاده از مدل‌های رگرسیون (به‌ویژه ریدج رگرسیون) بر مبنای داده‌های استخراج شده از پنجره‌های زمانی (sliding windows) است. در ادامه ابتدا بخش‌های مربوط به انتخاب ستون‌ها و دریافت اندیس‌های آن‌ها، سپس آموزش دو مدل با استفاده از دو نوع پنجره (یک پنجره بزرگ و یک پنجره یک‌روزه)، و در نهایت ترکیب نتایج و ارزیابی عملکرد مدل بیان خواهد شد.

۱. انتخاب ویژگی‌های مورد نظر و استخراج اندیس‌هایشان

در ابتدای کد، ستون‌های دیتافریم (new_df) که شامل عبارت‌های"temp" ، "wind"، "pressure"و "humidity"هستند جستجو می‌شوند. به‌عبارتی، با استفاده از متد str.contains ستون‌هایی را پیدا می‌کنیم که این کلمات را در نام خود دارند. به کمک متدtolist ، این نام‌ها به صورت یک لیست در می‌آیند. سپس، با استفاده از متد get_loc ، اندیس (شماره ستون) هر کدام از این ویژگی‌ها در دیتافریم اصلی استخراج می‌شود. این روند برای دسته‌بندی ویژگی‌های مرتبط با دما، باد، فشار و رطوبت انجام می‌شود تا بعدها بتوان از آن‌ها در انتخاب ویژگی‌های مفید برای آموزش مدل استفاده کرد.

Finding And extracting Required locations

Getting column indices for Temprature(T), Wind Spee(WS), Pressure(P) and Humidity(H)

```
#Finding Columns Including the wanted Features
t_cols = new_df.columns[new_df.columns.str.contains("temp", case=False, na=False)].tolist()
ws_cols=new_df.columns[new_df.columns.str.contains("wind", case=False, na=False)].tolist()
p_cols=new_df.columns[new_df.columns.str.contains("pressure", case=False, na=False)].tolist()
h_cols=new_df.columns[new_df.columns.str.contains("humidity", case=False, na=False)].tolist()
t_cols,ws_cols,p_cols,h_cols
```

```
(['MONTELIMAR_temp_mean',
  'MONTELIMAR_temp_min',
  'MONTELIMAR_temp_max',
  'PERPIGNAN_temp_mean',
  'PERPIGNAN_temp_min',
  'PERPIGNAN_temp_max',
  'TOURS_temp_mean',
  'TOURS_temp_min',
  'TOURS_temp_max'],
 ['MONTELIMAR_wind_speed', 'PERPIGNAN_wind_speed', 'TOURS_wind_speed'],
 ['MONTELIMAR_pressure', 'PERPIGNAN_pressure', 'TOURS_pressure'],
 ['MONTELIMAR_humidity', 'PERPIGNAN_humidity', 'TOURS_humidity'])
```

```
[283] #Getting Indx Numbers
t_indx = [new_df.columns.get_loc(col) for col in t_cols]
ws_indx=[new_df.columns.get_loc(col) for col in ws_cols]
p_indx=[new_df.columns.get_loc(col) for col in p_cols]
h_indx=[new_df.columns.get_loc(col) for col in h_cols]
t_indx,ws_indx,p_indx,h_indx
```

```
([5, 6, 7, 13, 14, 15, 21, 22, 23], [0, 8, 16], [2, 10, 18], [1, 9, 17])
```

۲ .تعریف مدل‌ها و آماده‌سازی پیش‌بینی‌ها

دو مدل ریدج رگرسیون با مقدار منظم‌سازی (α=0.01) تعریف شده‌اند:

model1t: برای آموزش مدل بر روی پنجره‌های بزرگ (large window).

model1t1day: برای آموزش مدل بر روی پنجره‌های یک‌روزه (۱-day window).

همچنین دو لیست خالی به نام‌های predictions و predictions1day برای ذخیره نتایج پیش‌بینی هر دو مدل ایجاد شده‌اند. این پیش‌بینی‌ها در طول حلقه‌های بعدی تکمیل و ذخیره می‌شوند.

۳ .آموزش مدل‌های رگرسیون با استفاده از پنجره‌های زمانی

در بخش آموزش مدل بزرگ، یک حلقه اجرا می‌شـــود که به ازای هر نمونه از داده‌های xtest (به جز آخرین نمونه) مراحل زیر را طی می‌کند:

ابتدا مدل model1t با استفاده از ویژگی‌های انتخاب‌شده (اندیس‌های [۵,۶,۷,۱۳,۱۴,۱۵,۲۱,۲۲,۲۳,۱,۹,۱۷]) از داده‌های پنجره i (بعد از reshape به شکل (۱-, ۱)) و با مقدار هدف ytest[i,5] (که به صورت یک آرایه تک‌عنصری reshape شده است) آموزش داده می‌شود.

سپس با استفاده از همان الگو، پیش‌بینی برای پنجره بعدی (i+1) انجام شده و مقدار پیش‌بینی شده (اولین عنصر آرایه خروجی مدل) به لیست predictions اضافه می‌شود. به عبارت دیگر، این مدل به ازای هر پنجره، داده‌های آن را برای آموزش مدل استفاده می‌کند و سپس برای پنجره بعدی پیش‌بینی می‌کند. در ادامه، به‌طور مشابه، مدل یک‌روزه (model1t1day) نیز روی داده‌های xtest1day و ytest1day آموزش داده می‌شود و نتایج پیش‌بینی در لیست predictions1day ذخیره می‌شود.

Ridge Regression for Temp (Using temp_min temp_max temp_mean and humidity of all cities)

```
[284] #MONTELIMAR_temp_mean training
      model1t=Ridge(alpha=0.01)
      model1t1day=Ridge(alpha=0.01)
      predictions=[]
      predictions1day=[]

      #Train model large window
      for i in range(0, xtest.shape[0]-1,1):

          model1t.fit(xtest[i,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1),ytest[i,5].reshape(-1))
          preds=model1t.predict(xtest[i+1,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1))
          predictions.append(preds[0])

      #Train model 1 day
      for i in range(0, xtest1day.shape[0]-1,1):

          model1t1day.fit(xtest1day[i,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1),ytest1day[i,5].reshape(-1))
          preds1day=model1t1day.predict(xtest1day[i+1,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1))
          predictions1day.append(preds1day[0])
```

۴. تبدیل پیش‌بینی‌ها به سری‌های pandas و ترکیب با مقادیر واقعی

پس از پایان آموزش، لیست‌های حاوی نتایج پیش‌بینی (برای هر دو مدل) به سری‌های pandas تبدیل می‌شوند. این سری‌ها با استفاده از اندیس‌های test_df (که به تعداد پیش‌بینی‌ها محدود شده‌اند) هماهنگ می‌شوند. سپس با استفاده از تابع pd.concat، دو DataFrame به دست می‌آید:

combined: شامل مقادیر واقعی مربوط به «MONTELIMAR_temp_mean» و پیش‌بینی‌های مدل پنجره بزرگ.

combined1day: شامل مقادیر واقعی و پیش‌بینی‌های مدل یک‌روزه.

این دو DataFrame به‌گونه‌ای نام‌گذاری شده‌اند تا به وضوح وضعیت واقعی و پیش‌بینی‌شده هر مدل را نشان دهند.

۵. تنظیم زمان‌بندی (Shift) و ارزیابی عملکرد مدل

در مرحله بعد، با استفاده از متد shift و اعمال تغییرات اندیس زمانی (با استفاده از pd.DateOffset)، داده‌های واقعی در هر DataFrame به‌گونه‌ای جابجا می‌شوند که نسبت به پیش‌بینی‌ها به درستی تراز (alignment) شوند. به عنوان مثال، در combined، با شیفت مقدار ۶-، داده‌های واقعی ۶ روز جلوتر قرار می‌گیرند؛ به همین ترتیب در combined1day با شیفت ۲- تنظیم می‌شوند. سپس داده‌های ناقص (NaN) حذف شده و دو معیار ارزیابی به نام‌های MAPE (Mean Absolute Percentage Error) و تفاوت (Diff) بین مقدار واقعی و پیش‌بینی شده محاسبه می‌شود.

```
# Convert predictions to a Series with matching index
preds_series = pd.Series(predictions, index=test_df.index[:len(predictions)])
preds1day_series = pd.Series(predictions1day, index=test_df.index[:len(predictions1day)])
# Combine with actual values
combined = pd.concat([test_df['MONTELIMAR_temp_mean'][:len(predictions)], preds_series], axis=1)
combined.columns = ["Actual_Large_Window", "Prediction_Large_Window"]

combined1day = pd.concat([test_df['MONTELIMAR_temp_mean'][:len(predictions1day)], preds1day_series], axis=1)
combined1day.columns = ["Actual_1Day_Window", "Prediction_1Day_Window"]

# Check the first few rows
combined["Actual_Large_Window"]=combined.shift(-6)["Actual_Large_Window"].copy()
combined.index = combined.index - pd.DateOffset(days=-6)
combined = combined.dropna()
combined["MAPE"]=mean_absolute_percentage_error(combined["Actual_Large_Window"],combined["Prediction_Large_Window"])
combined["Diff"]=combined["Actual_Large_Window"]-combined["Prediction_Large_Window"]

combined1day["Actual_1Day_Window"]=combined1day.shift(-2)["Actual_1Day_Window"].copy()
combined1day.index = combined1day.index - pd.DateOffset(days=-2)
combined1day = combined1day.dropna()
combined1day["MAPE"]=mean_absolute_percentage_error(combined1day["Actual_1Day_Window"],combined1day["Prediction_1Day_Window"])
combined1day["Diff"]=combined1day["Actual_1Day_Window"]-combined1day["Prediction_1Day_Window"]
```

نتایج برای دما به صورت زیر میباشد. همانگونه که مشاهده می‌شود، استفاده از پنجره بزرگتر موجب کاهش خطا نسبت به حالت ۱ روزه شده است:

combined

DATE	Actual_Large_Window	Prediction_Large_Window	MAPE	Diff
2009-01-07	-1.3	0.2	0.183994	-1.5
2009-01-08	-0.9	-1.3	0.183994	0.4
2009-01-09	1.2	-0.9	0.183994	2.1
2009-01-10	1.6	1.2	0.183994	0.4
2009-01-11	2.0	1.6	0.183994	0.4
...	...	...	...	...
2009-12-21	1.6	-2.0	0.183994	3.6
2009-12-22	9.4	1.6	0.183994	7.8
2009-12-23	7.8	9.4	0.183994	-1.6
2009-12-24	11.1	7.8	0.183994	3.3
2009-12-25	9.2	11.1	0.183994	-1.9

353 rows × 4 columns

Next steps: Generate code with combined | View recommended plots | New interactive sheet

[271] combined1day

DATE	Actual_1Day_Window	Prediction_1Day_Window	MAPE	Diff
2009-01-03	2.8	2.2	0.213235	0.6
2009-01-04	1.8	2.8	0.213235	-1.0
2009-01-05	-0.4	1.8	0.213235	-2.2
2009-01-06	0.2	-0.4	0.213235	0.6
2009-01-07	-1.3	0.2	0.213235	-1.5
...	...	...	...	...
2009-12-25	9.2	11.1	0.213235	-1.9
2009-12-26	4.0	9.2	0.213235	-5.2
2009-12-27	5.6	4.0	0.213235	1.6
2009-12-28	3.6	5.6	0.213235	-2.0
2009-12-29	9.4	3.6	0.213235	5.8

361 rows × 4 columns

در استفاده از Lasso Regression، کد و نتایج به صورت زیر میباشد:

Lasso Regression for Temp (Using temp_min temp_max temp_mean and humidity of all cities)

```
[286] from sklearn.linear_model import Lasso
```

```
#MONTELIMAR_temp_mean training
model1t=Lasso(alpha=0.01)
model1t1day=Lasso(alpha=0.01)
predictions=[]
predictions1day=[]

#Train model large window
for i in range(0, xtest.shape[0]-1,1):

    model1t.fit(xtest[i,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1),ytest[i,5].reshape(-1))
    preds=model1t.predict(xtest[i+1,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1))
    predictions.append(preds[0])

#Train model 1 day
for i in range(0, xtest1day.shape[0]-1,1):

    model1t1day.fit(xtest1day[i,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1),ytest1day[i,5].reshape(-1))
    preds1day=model1t1day.predict(xtest1day[i+1,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1))
    predictions1day.append(preds1day[0])
```

DATE				
2009-01-07	-1.3	0.2	0.183994	-1.5
2009-01-08	-0.9	-1.3	0.183994	0.4
2009-01-09	1.2	-0.9	0.183994	2.1
2009-01-10	1.6	1.2	0.183994	0.4
2009-01-11	2.0	1.6	0.183994	0.4
...	...	...	...	...
2009-12-21	1.6	-2.0	0.183994	3.6
2009-12-22	9.4	1.6	0.183994	7.8
2009-12-23	7.8	9.4	0.183994	-1.6
2009-12-24	11.1	7.8	0.183994	3.3
2009-12-25	9.2	11.1	0.183994	-1.9

353 rows × 4 columns

Next steps: Generate code with combined | View recommended plots | New interactive sheet

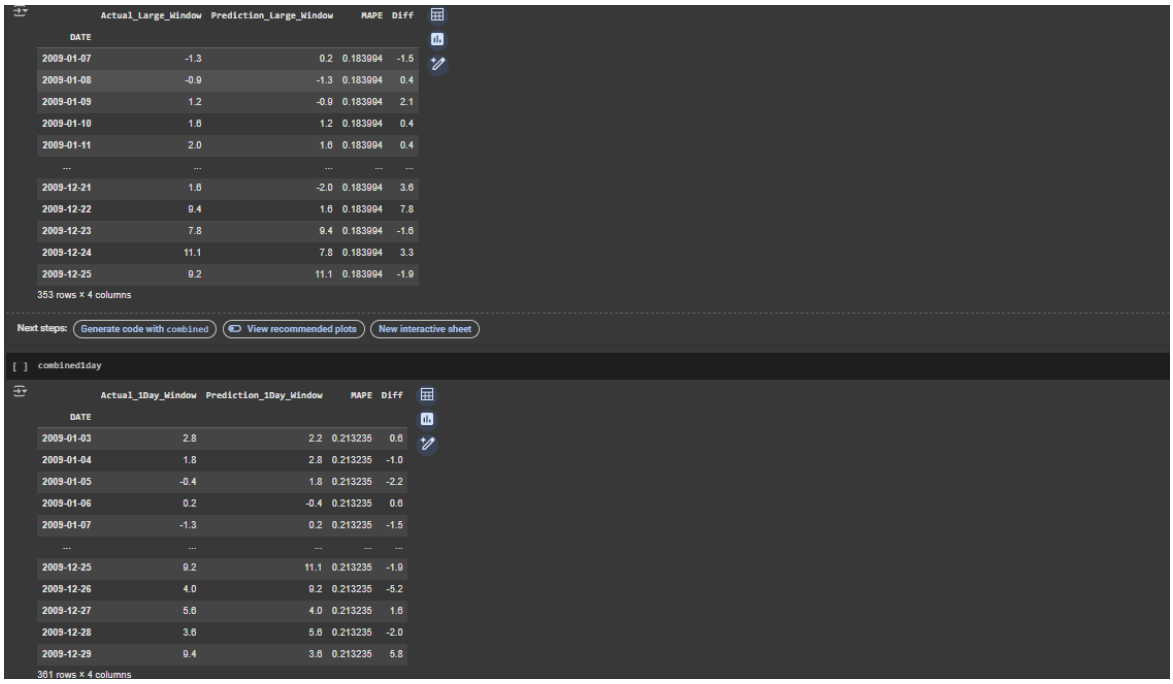
```
[ ] combined1day
```

DATE	Actual_1Day_Window	Prediction_1Day_Window	MAPE	Diff
2009-01-03	2.8	2.2	0.213235	0.6
2009-01-04	1.8	2.8	0.213235	-1.0
2009-01-05	-0.4	1.8	0.213235	-2.2
2009-01-06	0.2	-0.4	0.213235	0.6
2009-01-07	-1.3	0.2	0.213235	-1.5
...	...	...	...	...
2009-12-25	9.2	11.1	0.213235	-1.9
2009-12-26	4.0	9.2	0.213235	-5.2
2009-12-27	5.6	4.0	0.213235	1.6
2009-12-28	3.6	5.6	0.213235	-2.0
2009-12-29	9.4	3.6	0.213235	5.8

361 rows × 4 columns

به طور مشابه برای Linear Regression میتوان نوشت:

```python
#MONTELIMAR_temp_mean training
model1t=LinearRegression()
model1t1day=LinearRegression()
predictions=[]
predictions1day=[]

#Train model large window
for i in range(0, xtest.shape[0]-1,1):

    model1t.fit(xtest[i,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1),ytest[i,5].reshape(-1))
    preds=model1t.predict(xtest[i+1,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1))
    predictions.append(preds[0])

#Train model 1 day
for i in range(0, xtest1day.shape[0]-1,1):

    model1t1day.fit(xtest1day[i,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1),ytest1day[i,5].reshape(-1))
    preds1day=model1t1day.predict(xtest1day[i+1,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1))
    predictions1day.append(preds1day[0])
```



DATE	Actual_Large_Window	Prediction_Large_Window	MAPE	Diff
2009-01-07	-1.3	0.2	0.183994	-1.5
2009-01-08	-0.9	-1.3	0.183994	0.4
2009-01-09	1.2	-0.9	0.183994	2.1
2009-01-10	1.6	1.2	0.183994	0.4
2009-01-11	2.0	1.6	0.183994	0.4
...	...	...	...	...
2009-12-21	1.6	-2.0	0.183994	3.6
2009-12-22	9.4	1.6	0.183994	7.8
2009-12-23	7.8	9.4	0.183994	-1.6
2009-12-24	11.1	7.8	0.183994	3.3
2009-12-25	9.2	11.1	0.183994	-1.9

353 rows × 4 columns

Next steps: Generate code with combined | View recommended plots | New interactive sheet

```
[ ] combined1day
```

DATE	Actual_1Day_Window	Prediction_1Day_Window	MAPE	Diff
2009-01-03	2.8	2.2	0.213235	0.6
2009-01-04	1.8	2.8	0.213235	-1.0
2009-01-05	-0.4	1.8	0.213235	-2.2
2009-01-06	0.2	-0.4	0.213235	0.6
2009-01-07	-1.3	0.2	0.213235	-1.5
...	...	...	...	...
2009-12-25	9.2	11.1	0.213235	-1.9
2009-12-26	4.0	9.2	0.213235	-5.2
2009-12-27	5.6	4.0	0.213235	1.6
2009-12-28	3.6	5.6	0.213235	-2.0
2009-12-29	9.4	3.6	0.213235	5.8

361 rows × 4 columns

با توجه به نتایج بدست آمده، با توجه به داده‌های موجود و هدف پیش‌بینی دمای آینده، با اینکه نتایج تا حد خوبی شبیه میباشند ولی از آنجایی که نوع داده ها سری زمانی است، مدل Ridge Regression بهترین عملکرد را داشته است. این مدل با متعادل‌سازی ضرایب و جلوگیری از بیش‌برازش، دقت مناسبی را در عین پایداری ارائه کرده است.

سوال امتیازی ۱:

به طور مشابه مدل Ridge Regression برای Prepignan Temp mean نمایش داده شده است:

training for Prepignan Temp mean

```
#Prepignan training
model1t=Ridge(alpha=0.01)
model1t1day=Ridge(alpha=0.01)
predictions=[]
predictions1day=[]

#Train model large window
for i in range(0, xtest.shape[0]-1,1):

    model1t.fit(xtest[i,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1),ytest[i,13].reshape(-1))
    preds=model1t.predict(xtest[i+1,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1))
    predictions.append(preds[0])

#Train model 1 day
for i in range(0, xtest1day.shape[0]-1,1):

    model1t1day.fit(xtest1day[i,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1),ytest1day[i,13].reshape(-1))
    preds1day=model1t1day.predict(xtest1day[i+1,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1))
    predictions1day.append(preds1day[0])
```

```
combined
```

DATE	Actual_Large_Window	Prediction_Large_Window	MAPE	Diff
2009-01-07	1.4	4.0	0.151777	-2.6
2009-01-08	1.4	1.4	0.151777	0.0
2009-01-09	2.2	1.4	0.151777	0.8
2009-01-10	6.6	2.2	0.151777	4.4
2009-01-11	8.4	6.6	0.151777	1.8
...	...	...	...	...
2009-12-21	4.9	4.5	0.151777	0.4
2009-12-22	8.5	4.9	0.151777	3.6
2009-12-23	10.6	8.5	0.151777	2.1
2009-12-24	13.5	10.6	0.151777	2.9
2009-12-25	10.2	13.5	0.151777	-3.3

353 rows × 4 columns

Next steps: Generate code with combined | View recommended plots | New interactive sheet

```
[310] combined1day
```

DATE	Actual_1Day_Window	Prediction_1Day_Window	MAPE	Diff
2009-01-03	4.9	7.8	0.155633	-2.9
2009-01-04	7.0	4.9	0.155633	2.1
2009-01-05	6.4	7.0	0.155633	-0.6
2009-01-06	4.0	6.4	0.155633	-2.4
2009-01-07	1.4	4.0	0.155633	-2.6
...	...	...	...	...
2009-12-25	10.2	13.5	0.155633	-3.3
2009-12-26	7.8	10.2	0.155633	-2.4
2009-12-27	10.7	7.8	0.155633	2.9
2009-12-28	8.2	10.7	0.155633	-2.5
2009-12-29	9.5	8.2	0.155633	1.3

361 rows × 4 columns

برای شهر TOURS temp mean نیز به شکل زیر است:

Training for Tours temp mean

```
#Tours training
model1t=Ridge(alpha=0.01)
model1t1day=Ridge(alpha=0.01)
predictions=[]
predictions1day=[]

#Train model large window
for i in range(0, xtest.shape[0]-1,1):

    model1t.fit(xtest[i,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1),ytest[i,21].reshape(-1))
    preds=model1t.predict(xtest[i+1,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1))
    predictions.append(preds[0])

#Train model 1 day
for i in range(0, xtest1day.shape[0]-1,1):

    model1t1day.fit(xtest1day[i,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1),ytest1day[i,21].reshape(-1))
    preds1day=model1t1day.predict(xtest1day[i+1,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1))
    predictions1day.append(preds1day[0])
```

```
combined
```

DATE	Actual_Large_Window	Prediction_Large_Window	MAPE	Diff
2009-01-07	-6.2	-4.9	5.741133e+13	-1.3
2009-01-08	-1.4	-6.2	5.741133e+13	4.8
2009-01-09	-5.0	-1.4	5.741133e+13	-3.6
2009-01-10	-3.6	-5.0	5.741133e+13	1.4
2009-01-11	-1.8	-3.6	5.741133e+13	1.8
...	...	...	...	...
2009-12-21	4.8	0.0	5.741133e+13	4.8
2009-12-22	3.4	4.8	5.741133e+13	-1.4
2009-12-23	3.8	3.4	5.741133e+13	0.4
2009-12-24	4.8	3.8	5.741133e+13	1.0
2009-12-25	5.4	4.8	5.741133e+13	0.6

353 rows × 4 columns

Next steps: Generate code with combined | View recommended plots | New interactive sheet

```
[313] combined1day
```

DATE	Actual_1Day_Window	Prediction_1Day_Window	MAPE	Diff
2009-01-03	-0.1	-0.4	5.613905e+13	0.3
2009-01-04	-2.4	-0.1	5.613905e+13	-2.3
2009-01-05	-4.4	-2.4	5.613905e+13	-2.0
2009-01-06	-4.9	-4.4	5.613905e+13	-0.5
2009-01-07	-6.2	-4.9	5.613905e+13	-1.3
...	...	...	...	...
2009-12-25	5.4	4.8	5.613905e+13	0.6
2009-12-26	2.6	5.4	5.613905e+13	-2.8
2009-12-27	4.6	2.6	5.613905e+13	2.0
2009-12-28	6.2	4.6	5.613905e+13	1.6
2009-12-29	10.4	6.2	5.613905e+13	4.2

361 rows × 4 columns

سوال امتیازی ۲

دیتا فریم new_df2 تنها به منظور استفاده در این بخش ایجاد شده و به طور مشابه بخش های قبل، موارد مورد نیاز برای آن استخراج گردیده است.

از شهرهای DUSSELDORF, DRESDEN, KASSEL برای انجام پیش بینی در شهر DUSSELDORF (همگی واقع در آلمان) استفاده شده است.

```
  # THIS PART IS ONLY DEFINED FOR BOUNUS QUESTION 2 AND NOT USED ANYWHERE ELSE !!!!!
search_columns2 = [col for col in weather_prediction_dataset.columns \
   if any(keyword in col for keyword in ["DUSSELDORF", "DRESDEN", "KASSEL"])]
# extracts only the columns whose headers contain Dusseldorf, Dresden or kassel
new_df2=weather_prediction_dataset[search_columns2]
search_columns2
```

```
['DRESDEN_cloud_cover',
 'DRESDEN_wind_speed',
 'DRESDEN_wind_gust',
 'DRESDEN_humidity',
 'DRESDEN_global_radiation',
 'DRESDEN_precipitation',
 'DRESDEN_sunshine',
 'DRESDEN_temp_mean',
 'DRESDEN_temp_min',
 'DRESDEN_temp_max',
 'DUSSELDORF_cloud_cover',
 'DUSSELDORF_wind_speed',
 'DUSSELDORF_wind_gust',
 'DUSSELDORF_humidity',
 'DUSSELDORF_pressure',
 'DUSSELDORF_global_radiation',
 'DUSSELDORF_precipitation',
 'DUSSELDORF_sunshine',
 'DUSSELDORF_temp_mean',
 'DUSSELDORF_temp_min',
 'DUSSELDORF_temp_max',
 'KASSEL_wind_speed',
 'KASSEL_wind_gust',
 'KASSEL_humidity',
 'KASSEL_pressure',
 'KASSEL_global_radiation',
 'KASSEL_precipitation',
 'KASSEL_sunshine',
 'KASSEL_temp_mean',
 'KASSEL_temp_min',
 'KASSEL_temp_max']
```

سوال امتیازی دوم

predictions for Dusseldorf using Dusseldorf, Dresden and Kassel

```
#Finding Columns Including the wanted Features
t_colss = new_df2.columns[new_df2.columns.str.contains("temp", case=False, na=False)].tolist()
ws_colss=new_df2.columns[new_df2.columns.str.contains("wind", case=False, na=False)].tolist()
p_colss=new_df2.columns[new_df2.columns.str.contains("pressure", case=False, na=False)].tolist()
h_colss=new_df2.columns[new_df2.columns.str.contains("humidity", case=False, na=False)].tolist()
t_colss,ws_colss,p_colss,h_colss
```

```
(['DRESDEN_temp_mean',
  'DRESDEN_temp_min',
  'DRESDEN_temp_max',
  'DUSSELDORF_temp_mean',
  'DUSSELDORF_temp_min',
  'DUSSELDORF_temp_max',
  'KASSEL_temp_mean',
  'KASSEL_temp_min',
  'KASSEL_temp_max'],
 ['DRESDEN_wind_speed',
  'DRESDEN_wind_gust',
  'DUSSELDORF_wind_speed',
  'DUSSELDORF_wind_gust',
  'KASSEL_wind_speed',
  'KASSEL_wind_gust'],
 ['DUSSELDORF_pressure', 'KASSEL_pressure'],
 ['DRESDEN_humidity', 'DUSSELDORF_humidity', 'KASSEL_humidity'])
```

```
#Getting Indx Numbers
t_indxx = [new_df2.columns.get_loc(col) for col in t_colss]
ws_indxx=[new_df2.columns.get_loc(col) for col in ws_colss]
p_indxx=[new_df2.columns.get_loc(col) for col in p_colss]
h_indxx=[new_df2.columns.get_loc(col) for col in h_colss]
t_indxx,ws_indxx,p_indxx,h_indxx
```

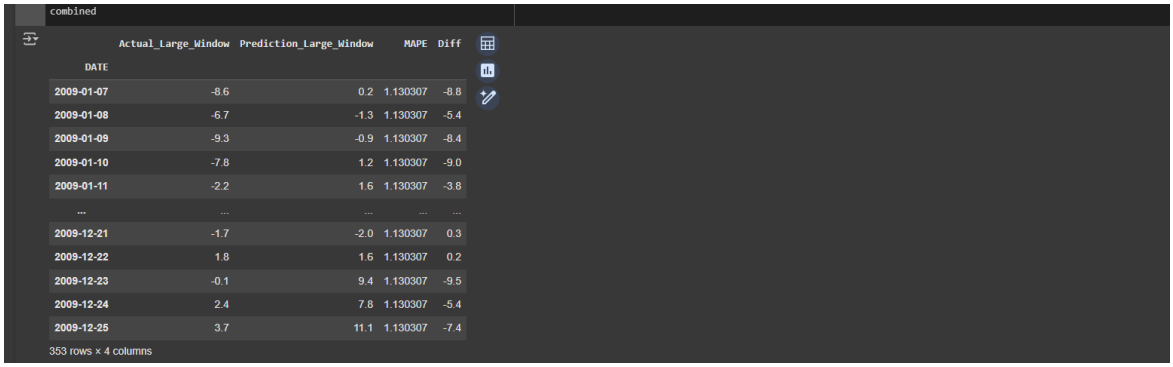
```
([7, 8, 9, 18, 19, 20, 28, 29, 30],
 [1, 2, 11, 12, 21, 22],
 [14, 24],
 [3, 13, 23])
```

```
test_df2=new_df2[new_df2.index >= '2009-01-01'].copy()
#cutting 2010
test_df2=test_df2.iloc[:-1,:].copy()
xtest2,ytest2=Create_Sliding_Windows(test_df2, window_size, step)
xtest2.shape, xtest.shape
```

```
((360, 5, 31), (360, 5, 24))
```

```
model1t=Ridge(alpha=0.01)
predictions=[]

#Train model large window
for i in range(0, xtest2.shape[0]-1,1):

    model1t.fit(xtest[i,:,[5,6,7,13,14,15,21,22,23,1,9,17]].reshape(1,-1),ytest[i,5].reshape(-1))
    preds=model1t.predict(xtest2[i+1,:,[7,8,9,18,19,20,28,29,30,3,13,23]].reshape(1,-1))
    predictions.append(preds[0])


# Convert predictions to a Series with matching index
preds_series = pd.Series(predictions, index=test_df.index[:len(predictions)])

# Combine with actual values
combined = pd.concat([test_df2['DUSSELDORF_temp_mean'][:len(predictions)], preds_series], axis=1)
combined.columns = ["Actual_Large_Window", "Prediction_Large_Window"]


# Check the first few rows
combined["Actual_Large_Window"]=combined.shift(-6)["Actual_Large_Window"].copy()
combined.index = combined.index - pd.DateOffset(days=-6)
combined = combined.dropna()
combined["MAPE"]=mean_absolute_percentage_error(combined["Actual_Large_Window"],combined["Prediction_Large_Window"])
combined["Diff"]=combined["Actual_Large_Window"]-combined["Prediction_Large_Window"]



combined
```

در اینجا نیز مانند بخش‌های قبل، ابتدا پنجره اول برای آموزش استفاده شده ولی با این تفاوت که پنجره دوم از شهر دوسلدورف به عنوان ورودی میباشد. نتایج به شکل زیر است:



combined

DATE	Actual_Large_Window	Prediction_Large_Window	MAPE	Diff
2009-01-07	-8.6	0.2	1.130307	-8.8
2009-01-08	-6.7	-1.3	1.130307	-5.4
2009-01-09	-9.3	-0.9	1.130307	-8.4
2009-01-10	-7.8	1.2	1.130307	-9.0
2009-01-11	-2.2	1.6	1.130307	-3.8
...	...	...	...	...
2009-12-21	-1.7	-2.0	1.130307	0.3
2009-12-22	1.8	1.6	1.130307	0.2
2009-12-23	-0.1	9.4	1.130307	-9.5
2009-12-24	2.4	7.8	1.130307	-5.4
2009-12-25	3.7	11.1	1.130307	-7.4

353 rows × 4 columns

همانگونه که مشاهده می‌شود، با اینکه بین مقادیر پیش بینی شده و واقعی اختلاف وجود دارد ولی این خطا تا حدودی قابل قبول میباشد زیرا با توجه به مبحث یادگیری انتقالی اگر داده‌های آب‌وهوا (مثلاً دما، فشار، رطوبت، باد و ...) در شهرهای مختلف دارای توزیع‌های آماری مشابه باشند، مدل آموزش‌دیده قادر خواهد بود تا اطلاعات کسب‌شده را به‌خوبی به کار بگیرد و عملکرد نسبتاً مناسبی در شهرهای دیگر ارائه دهد. اما اگر تفاوت قابل توجهی در الگوهای آب‌وهوا یا شرایط جوی بین شهرها وجود داشته باشد، ممکن است عملکرد مدل کاهش یابد.

همچنین اگر ویژگی‌هایی که انتخاب شده‌اند، شاخص‌های عمومی و اساسی وضعیت آب‌وهوا باشند و به‌طور گسترده در شهرها مشترک باشند، احتمال انتقال دانش از یک شهر به دیگری بیشتر می‌شود. در این صورت، مدل در ارزیابی روی داده‌های شهر دیگر می‌تواند پیش‌بینی‌های قابل قبولی ارائه دهد. بنابراین، در صورت استفاده از ویژگی‌های یکسان استخراج‌شده از سه شهر دیگر و ارزیابی مدل آموزش‌دیده بر روی داده‌های یکی از آن شهرها، اگر میان داده‌های شهرها شباهت و همگنی وجود داشته باشد، عملکرد مدل به‌طور قابل قبولی حفظ خواهد شد (اگرچه ممکن است مقداری افت در دقت مشاهده شود). اما اگر اختلافات قابل توجهی بین الگوهای آب‌وهوا در شهرها وجود داشته باشد، مدل ممکن است عملکرد ضعیف‌تری نشان دهد زیرا در اصل در محیط هدف با چالش‌هایی مانند عدم تطابق توزیع داده‌ها مواجه می‌شود.

به بیان خلاصه، در شرایط مطلوب (شباهت در داده‌های ورودی بین شهرها و ویژگی‌های استخراج‌شده‌ای که به طور گسترده نمایانگر شرایط آب‌وهوا هستند)، مدل انتقالی عملکرد نسبتاً قابل قبولی خواهد داشت، ولی در شرایط ناسازگار، کاهش قابل ملاحظه‌ای در دقت پیش‌بینی اتفاق می‌افتد.

۲. تشخیص عیب یاتاقان غلتشی بر مبنای دسته‌بندی‌های سلسله مراتبی

توضیحات مربوط به کد، به صورت نظر در خطوط کد نوشته شده موجود است.

1.1.2

سازوکار کلی داده برداری:

-دستگاه استفاده شده: شبیه ساز عیب ماشین آلات SpectraQuest (مدل Alignment-Balance-Vibration) برای شبیه سازی عیبهای مکانیکی در محیط کنترل شده.

- نرخ نمونه برداری: ۵۰ کیلوهرتز (۵۰,۰۰۰ نمونه در ثانیه).

- مدت زمان جمع آوری داده: ۵ ثانیه برای هر آزمایش.

- تعداد نمونه ها: ۲۵۰,۰۰۰ نمونه در هر آزمایش (نتیجه ۵۰,۰۰۰ نمونه/ثانیه × ۵ ثانیه).

- فرمت ذخیرهسازی: دادهها در فایلهای CSV ذخیره میشوند و هر ستون مربوط به یک سنسور یا پارامتر خاص است.

سنسورهای استفاده شده:

۱. سرعت سنج:

- مدل: Monarch Instrument MT-190 (آنالوگ).

- نقش: اندازه گیری سرعت چرخش روتور.

- ذخیره سازی: دادهها در ستون اول فایلهای CSV.

۲. شتاب سنج ها:

- یاتاقان Underhang (زیرین شفت روتور):

- مدل: سه شتابسنج IMI Sensors 601A012.

- نقش: اندازه گیری ارتعاش در سه جهت:

- جهت شعاعی (ستون ۲).

- جهت محوری (ستون ۳)

- جهت مماسی (ستون ۴).

- یاتاقان Overhang (جلویی شفت روتور):

- مدل: یک شتابسنج سه محوری IMI Sensors 604B31.

- نقش: اندازه گیری ارتعاش در سه جهت (شعاعی، محوری، مماسی).

- ذخیرهسازی: دادهها در ستونهای ۵ تا ۷ فایلهای CSV.

۳. میکروفون:

- مدل: Shure SM81.

- نقش: ثبت سیگنالهای صوتی ناشی از شرایط عیب.

- ذخیرهسازی: دادهها در ستون ۸ فایلهای CSV

اهداف داده برداری:

- ثبت داده های با وضوح بالا برای تحلیل رفتارهای گذرا (مانند شروع چرخش) و پایدار ماشین تحت شرایط مختلف عیب.

- شبیه سازی دقیق عیب های مکانیکی (مانند خرابی یاتاقانها) بدون آسیب به تجهیزات واقعی.

- ایجاد پایگاه دادهای جامع برای توسعه الگوریتمهای تشخیص عیب و استراتژیهای نگهداری پیش بینانه.

2.1.2

کلاس های مختلف عیب در مجموعه داده MaFaulDa:

۱. ناهماهنگی (Misalignment):

- ناهماهنگی افقی: با سطوح شدت از ۰.۵۰ میلیمتر تا ۲.۰۰ میلیمتر.

- ناهماهنگی عمودی: با سطوح شدت از ۰.۵۱ میلیمتر تا ۱.۹۰ میلیمتر.

این عیب ها با جابه جایی محور روتور نسبت به محور موتور شبیه سازی میشوند و منجر به ارتعاشات غیرعادی در سیستم میگردند.

۲. عدم تعادل (Imbalance):

- سطوح شدت: اضافه کردن وزنهای ۶ گرم تا ۳۵ گرم به روتور.

این عیب ناشی از توزیع نابرابر جرم در اطراف محور روتور است که باعث ایجاد نیروی گریز از مرکز و افزایش ارتعاشات میشود.

۳. عیب های یاتاقان (Bearing Faults):

- عیب قفس (Cage Defect): اعمال وزنهای ۰ گرم تا ۳۵ گرم روی هر دو یاتاقان Overhang و Underhang.

- عیب بیرونی (Outer Race Defect): اعمال وزنهای ۰ گرم تا ۳۵ گرم روی هر دو یاتاقان.

- عیب توپ (Ball Defect): ایجاد آسیب در عناصر غلتشی یاتاقانها با سطوح شدت ۰ گرم تا ۳۵ گرم.

این عیبها بهصورت مصنوعی روی یاتاقانها ایجاد میشوند تا شرایط خرابی واقعی (مانند ترک یا ساییدگی) را شبیهسازی کنند.

مقدار داده ثبتشده برای هر بخش:

- نرخ نمونهبرداری: ۵۰ کیلوهرتز (۵۰,۰۰۰ نمونه در ثانیه).

- مدت زمان هر آزمایش: ۵ ثانیه.

- تعداد نمونه ها در هر آزمایش:

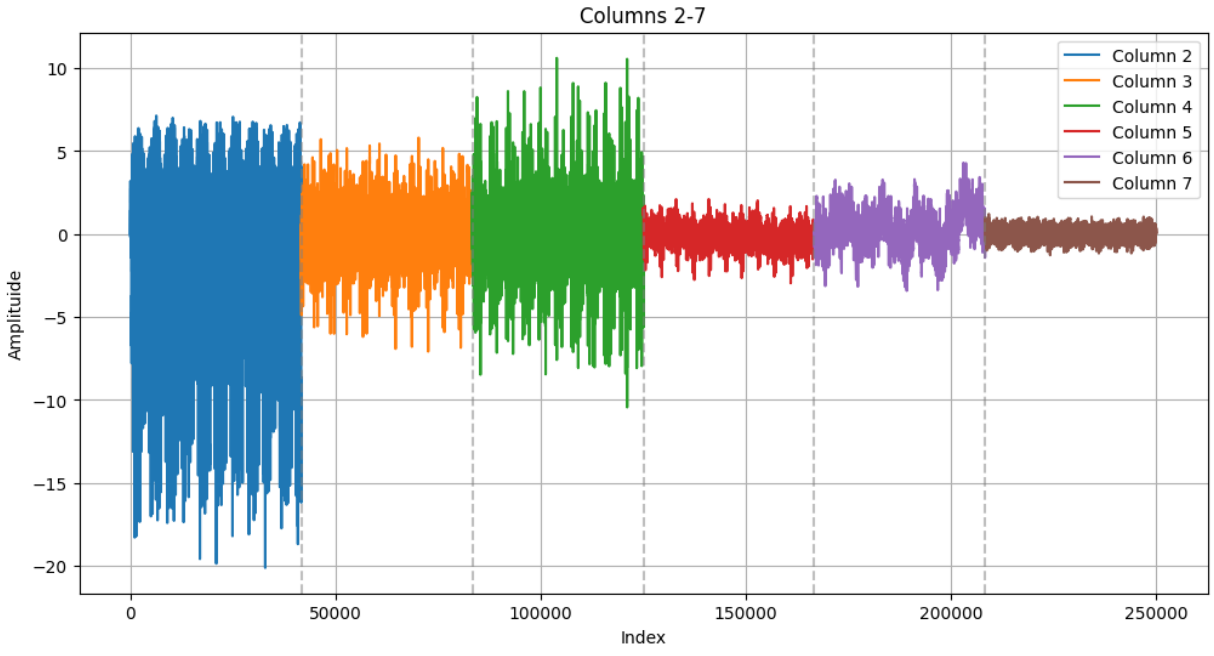
۲۵۰,۰۰۰ نمونه (حاصل از ۵۰,۰۰۰ نمونه/ثانیه × ۵ ثانیه).

جمع بندی:

هدف: ایجاد یک پایگاه داده جامع برای تحلیل رفتار ماشین تحت شرایط عیب و توسعه مدلهای تشخیص عیب.

پوشش عیب ها: مجموعه داده شامل ترکیبی از عیبهای ناهماهنگی، عدم تعادل، و یاتاقان (با زیرمجموعه-های قفس، بیرونی، و توپ) است که هر یک در سطوح شدت متفاوت شبیه‌سازی شده اند.

3.1.2



1.2.2

مرحله استخراج ویژگی از انتخاب پنجره زمانی (Time Windowing) شروع شده است. این مرحله به عنوان نخستین گام در فرآیند استخراج ویژگیها ذکر شده و هدف آن تقسیم سیگنالهای خام به بازههای زمانی مشخص (پنجره ها) برای تحلیل دقیق تر است. انتخاب پنجره زمانی، پایه و اساس استخراج ویژگیها در حوزه زمان و فرکانس را فراهم میکند.

پیش پردازش های انجام شده روی داده ها

۱. تقسیم سیگنال به پنجره های زمانی:

- تقسیم سیگنال های ارتعاشی و صوتی به بازه های زمانی کوتاه تر (پنجره ها) برای تحلیل موضعی.

- تنظیم طول پنجره و همپوشانی (Overlap) بر اساس:

- ماهیت سیگنال (ثابت/غیرثابت).

- نوع عیب (مثلاً پنجرههای کوتاهتر برای عیبهای ضربهای مانند خرابی یاتاقان).

- اهداف تحلیل (تشخیص عیب vs. مانیتورینگ وضعیت).

۲. تبدیل سیگنال به حوزه فرکانس:

- استفاده از تبدیل فوریه سریع (FFT) برای تبدیل سیگنال های حوزه زمان به حوزه فرکانس.

- این تبدیل برای استخراج ویژگیهای طیفی (مانند دامنه فرکانسهای غالب، انرژی باندهای فرکانسی، و غیره) ضروری است.

۳. برچسب گذاری داده ها:

- مرتبط سازی هر پنجره زمانی با نوع عیب و سطوح شدت مربوطه (مثلاً ناهماهنگی ۰.۵ میلیمتر، عدم تعادل ۱۰ گرم، یا خرابی قفس یاتاقان).

۴. ساختاردهی دادهها برای تحلیل:

- سازماندهی داده ها در قالب ستون های مجزا برای هر سنسور (مثلاً ستون ۲ تا ۴ برای شتاب سنج های یاتاقان Underhang).

- آماده سازی داده ها برای ورود به مرحله استخراج ویژگی (مانند ذخیره سازی مقادیر پنجره ها در ماتریس های قابل پردازش).

2.2.2

در فایل کد انجام شده است. از LightGBM نیز برای رتبه بندی ویژگی ها و SI جهت نرمال سازی استفاده شده است.

3.2

از روش Hierarchical Logistic Regression استفاده شده است.

1.3.2

در پایان نامه از مدل سلسله مراتبی با ۱۲ طبقه بند تخصصی استفاده شده است. هر طبقه بند روی یک دسته خاص از عیوب (مانند ناهماهنگی افقی/عمودی، عدم تعادل، یا عیوب یاتاقان) تمرکز دارد. ساختار بهصورت زیر است:

- ۱ طبقه بند اصلی: تشخیص سالم/معیوب بودن سیستم.

- ۵ طبقه بند سطح میانی: شناسایی ۵ عیب کلی (ناهماهنگی افقی، عمودی، عدم تعادل، و عیوب یاتاقان Overhang/Underhang).

- ۶ طبقه بند سطح پایین: تشخیص سطوح شدت عیوب یاتاقان (Cage، Ball، Outer Race برای هر یاتاقان).

نحوه استفاده:

- هر طبقه بند به صورت مجزا آموزش داده شده و ویژگی های مرتبط با عیب خاص (زمانی/فرکانسی) را تحلیل میکند.

- ابتدا طبقه بند اصلی سالم/معیوب را تشخیص میدهد. سپس طبقه بندهای میانی و پایین، نوع و شدت عیب را شناسایی میکنند.

مزیت های ذکرشده:

۱. کاهش پیچیدگی: تقسیم مسئله ۴۲ کلاس به زیرمسائل کوچکتر با تمرکز روی ویژگیهای خاص هر عیب.

۲. افزایش دقت: استفاده از مدل های تخصصی برای هر عیب، به جای یک مدل کلی.

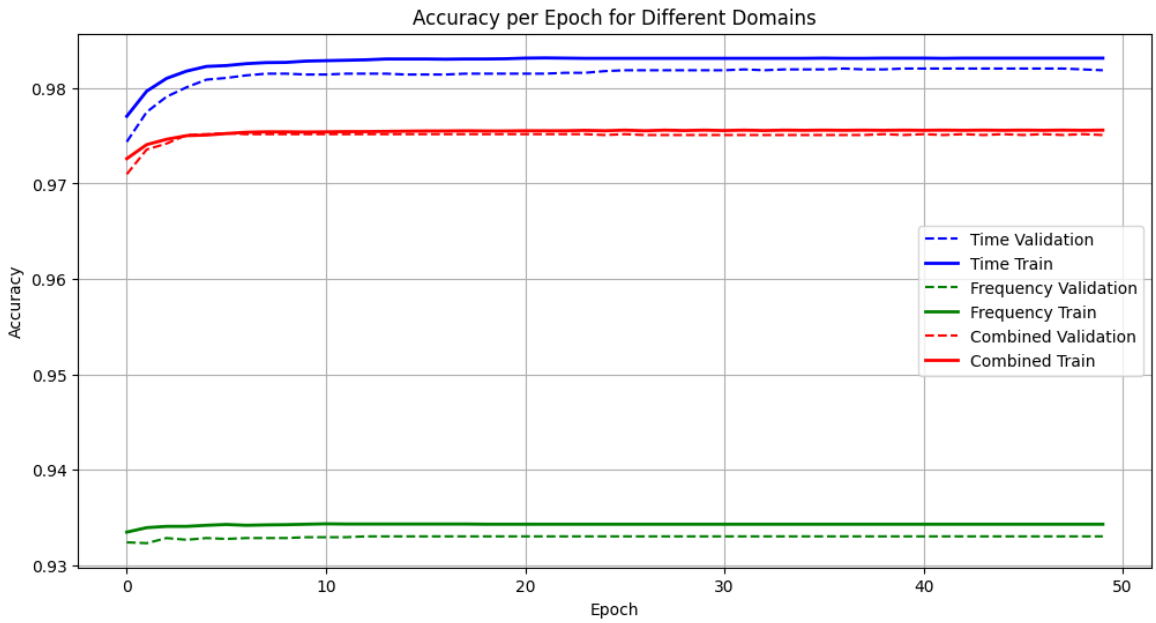
۳. بهبود تفسیرپذیری: ردیابی دقیق تر علت عیب در سطوح مختلف سلسله مراتبی.

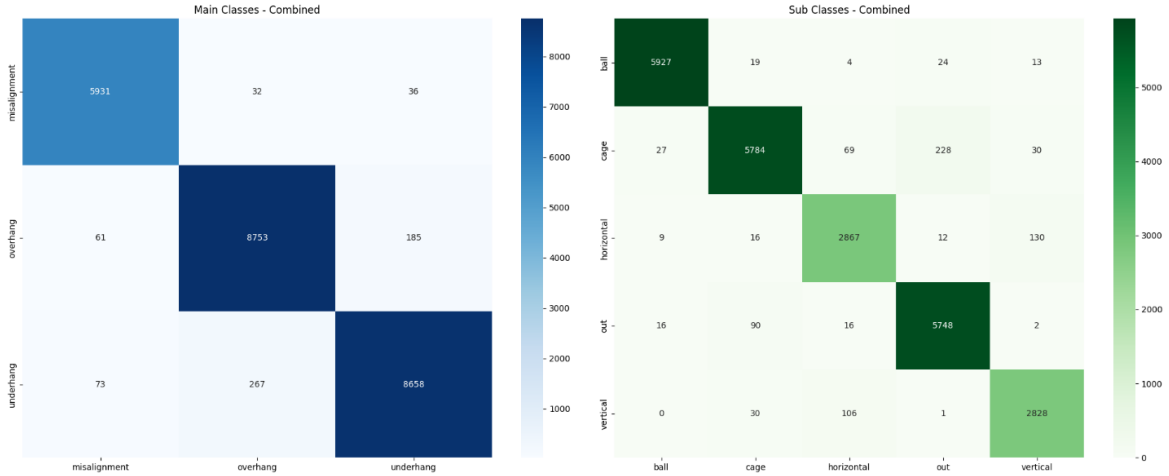
نتایج نهایی:

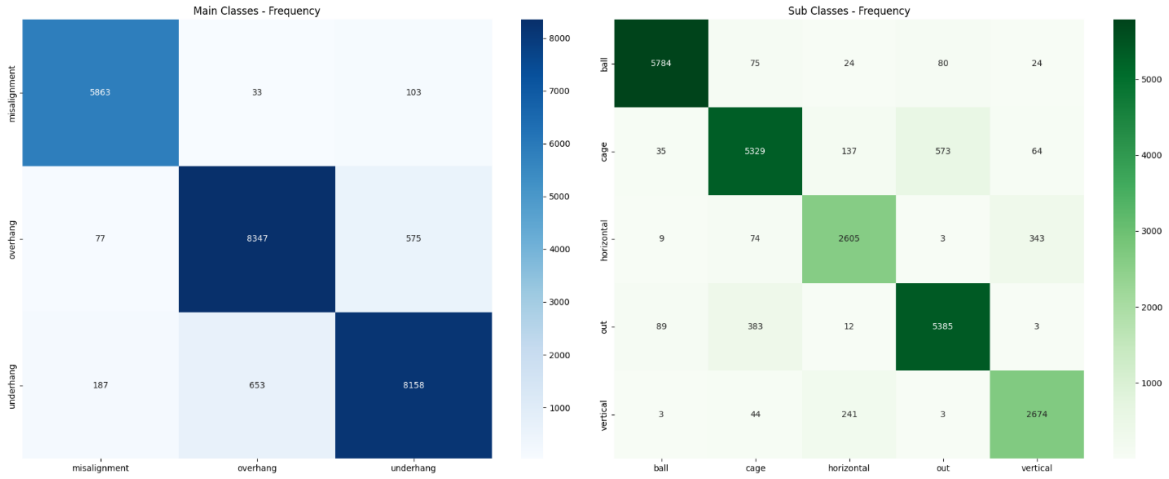
نویسنده تأکید میکند این روش به دلیل تمرکز روی ویژگی های مرتبط با هر عیب و کاهش ابعاد داده، دقت تشخیص را افزایش داده است. اگرچه نتایج کمی (مانند Accuracy) در متن ذکر نشده، اما منطق طراحی و تقسیم مسئله نشان دهنده تأثیر مثبت این روش است.

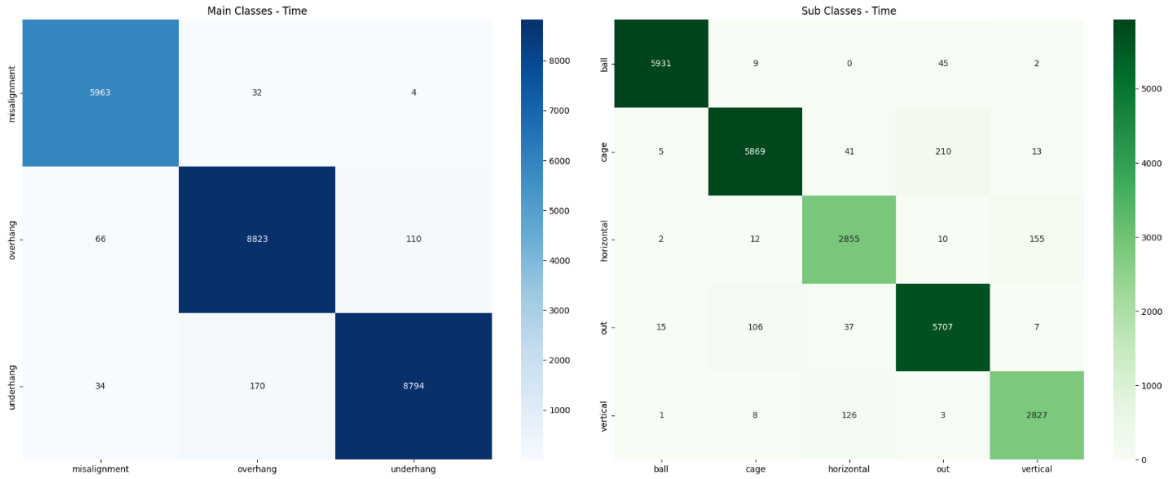
باتوجه به موارد فوق بنظر می رسد روش مطرح شده، روش قابل قبولی است و می تواند مفید باشد.

2.3.2 و 3.3.2

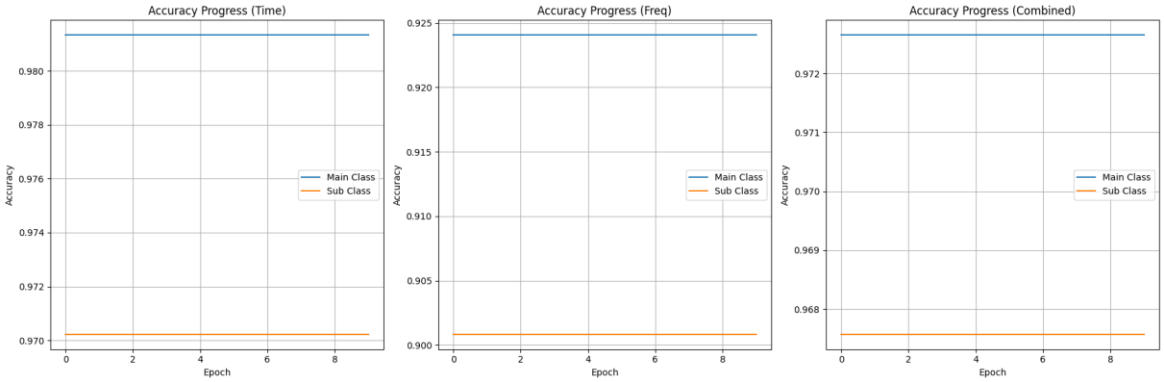


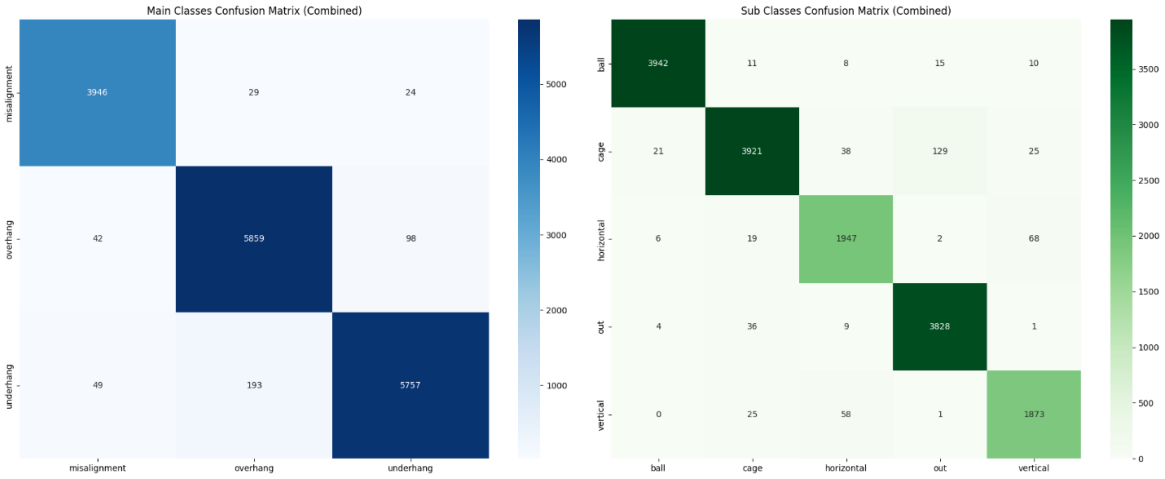

Main Classes - Combined
misalignment
overhang
underhang
5931
32
36
61
8753
185
73
267
8658
8000
7000
6000
5000
4000
3000
2000
1000
misalignment
overhang
underhang
Sub Classes - Combined
ball
cage
horizontal
out
vertical
5927
19
4
24
13
27
5784
69
228
30
9
16
2867
12
130
16
90
16
5748
2
0
30
106
1
2828
5000
4000
3000
2000
1000
0
ball
cage
horizontal
out
vertical

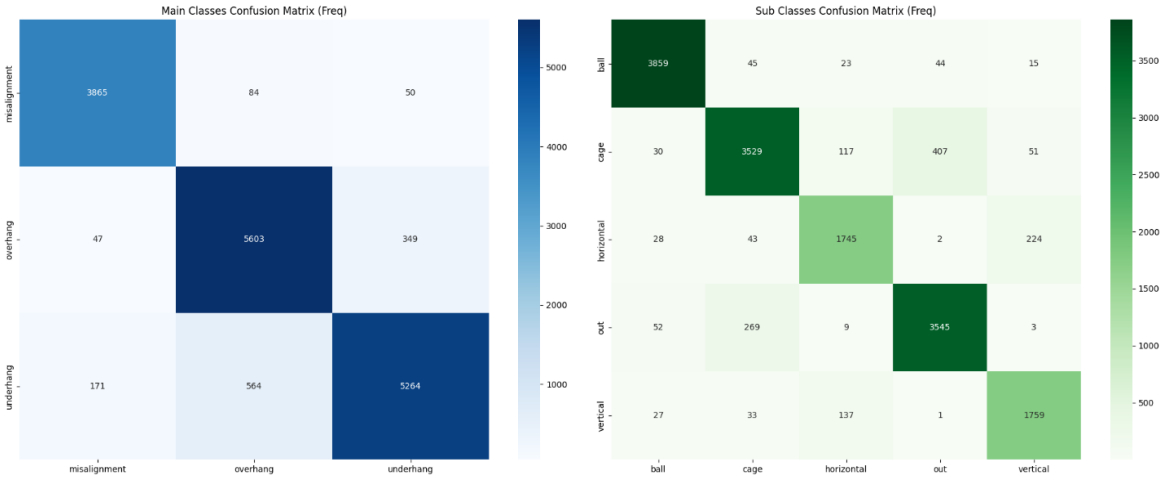

Main Classes - Frequency
misalignment
overhang
underhang
5863
33
103
77
8347
575
187
653
8158
8000
7000
6000
5000
4000
3000
2000
1000
misalignment
overhang
underhang
Sub Classes - Frequency
ball
cage
horizontal
out
vertical
5784
75
24
80
24
35
5329
137
573
64
9
74
2605
3
343
89
383
12
5385
3
3
44
241
3
2674
5000
4000
3000
2000
1000
ball
cage
horizontal
out
vertical

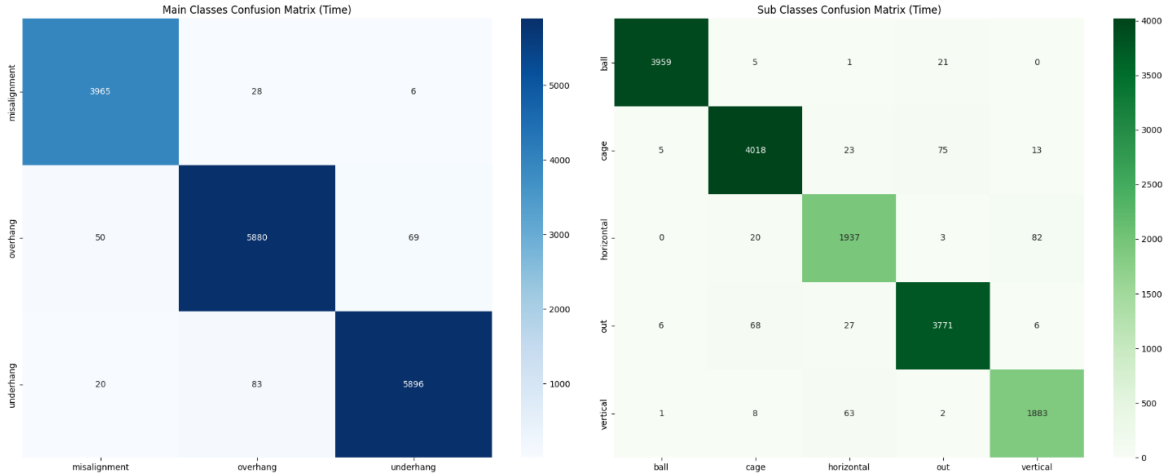

Main Classes - Time
misalignment
overhang
underhang
5963
32
4
66
8823
110
34
170
8794
8000
7000
6000
5000
4000
3000
2000
1000
misalignment
overhang
underhang
Sub Classes - Time
ball
cage
horizontal
out
vertical
5931
9
0
45
2
5
5869
41
210
13
2
12
2855
10
155
15
106
37
5707
7
1
8
126
3
2827
5000
4000
3000
2000
1000
0
ball
cage
horizontal
out
vertical

2.3.2 رویکرد Linear SVM









3.3.2

مقایسه دو روش

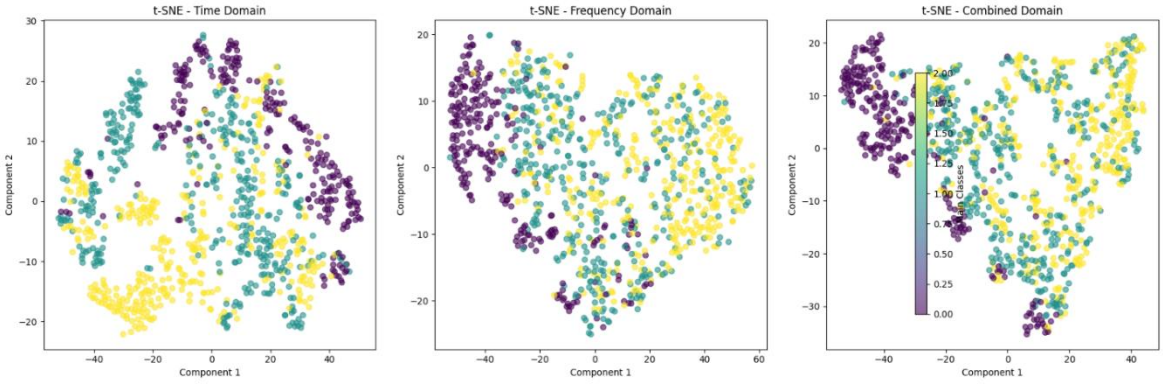
نتایج دو روش تقریبا برابر هستند و تفاوتی چندان در اینکه ویژگی های چه حوزه ای نیز استفاده شده است فرقی ندارند.

4.3.2

باتوجه به نتایج، روش پایان نامه پیشنهاد می شود چرا که مدت زمان نیاز برای اجرا آن بسیار کمتر است. اجرا برنامه برای Linear SVM برای ۱۰ epoch و ۵۰ epoch روش پایان نامه با هم برابر است.

4.2

t-SNEیک روش ساده برای کاهش ابعاد داده ها و نمایش آنها در فضای ۲ یا ۳ بعدی است. هدف اصلی آن این است که داده های پیچیده و با ابعاد زیاد (مثلاً در ۵۰۰ بعد) را به شکلی قابل درک مثلاً روی یک صفحه نشان دهد تا الگوها، خوشهها یا شباهت ها دیده شوند.


t-SNE - Time Domain
t-SNE - Frequency Domain
t-SNE - Combined Domain
Component 1
Component 2
Human Classes