به نام خدا


دانشگاه صنعتی خواجه نصیرالدین


دانشکده برق

# تشخیص و شناسایی خطا

## گزارش تمرین شماره ۳

**شیما سادات ناصری**

**۴۰۱۱۲۸۱۴**

**دکتر مهدی علیاری شوره دلی**

خرداد ماه ۱۴۰۲

# فهرست مطالب

عنوان شماره صفحه

## چکیده

در ۵ الی ۱۰ خط هدف از انجام این تمرین و آنچه را که خواهید آموخت به طور مختصر و مفید بنویسید.

## بخش۱: بررسی داده

بعد از بررسی‌هایی که در سایت‌های مختلف برای یک دیتاست مناسب برای ادامه پروژه انجام شد، دیتاست SIMULATED BOILER DATA FOR FAULT DETECTION AND CLASSIFICATION انتخاب شدند. این داده ها بر اساس یک شبیه‌ساز برای بویلر گاز سوز Viessmann Vitorond 200 VD2 Series 380 در Matlab/Simulink بر اساس مدل بویلر مجازی برپایه Simscape بدست آمده است. پس از تأیید اعتبار با داده های آزمایش سازنده، یک سری خطا (هوای اضافی، رسوب مبدل حرارتی، مقیاس بندی عنصر مبدل حرارتی سمت آب و شرایط احتراقی lean در بویلر) با این شبیه ساز، همراه با شرایط عملیاتی اسمی به عنوان داده نرمال برای طیف وسیعی از پارامترهای عملیاتی (نرخ سوخت گاز از ۱ کیلوگرم بر ثانیه تا ۴ کیلوگرم بر ثانیه، سرعت جریان جرمی آب از ۳ کیلوگرم بر ثانیه تا ۱۲.۵ کیلوگرم بر ثانیه و دمای هوای احتراق از ۲۸۳ کلوین تا ۳۰۳ کلوین) ارائه شده است. از روش نمونه گیری فاکتوریل استفاده شد که در مجموع ۲۷۲۸۱ شبیه سازی انجام شد. این مجموعه داده شامل نقاط قابل مشاهده برای یک سیستم اتوماسیون ساختمان به همراه Condition (کلاس تفصیلی) و Class (برچسب) است.

مشکل مهمی که این داده برای ما در این قسمت دارد این است که دارای داده‌های غیر عددی یا اسمی است. این داده‌ها در دو بخش تفصیلی و برچسب هستند. کلیات آن‌ها به صورت زیر می‌باشد.

```
1  data=pd.read_csv('data/Boiler_emulator_dataset.txt',delimiter=",")
2  data.head()
```
✓ 0.0s

| | Fuel_Mdot | Tair | Treturn | Tsupply | Water_Mdot | Condition | Class |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 283 | 333.0 | 363.574744 | 3.0 | %=0.05 | Lean |
| 1 | 1 | 283 | 333.0 | 362.349517 | 3.0 | %=0.1 | Nominal |
| 2 | 1 | 283 | 333.0 | 361.216941 | 3.0 | %=0.15 | ExcessAir |
| 3 | 1 | 283 | 333.0 | 360.166890 | 3.0 | %=0.20 | ExcessAir |
| 4 | 1 | 283 | 333.0 | 359.190662 | 3.0 | %=0.25 | ExcessAir |

```
1  pd.unique(data['Class'])
```
✓ 0.0s

```
array(['Lean', 'Nominal', 'ExcessAir', 'Fouling', 'Scaling'], dtype=object)
```

```
1  pd.unique(data['Condition'])
```
✓ 0.0s

```
array(['%=0.05', '%=0.1', '%=0.15', '%=0.20', '%=0.25', '%=0.3', '%=0.35',
       '%=0.40', '%=0.45', '%=0.50', 'F = 0.01', 'F = 0.06', 'F = 0.11',
       'F = 0.16', 'F = 0.21', 'F = 0.26', 'F = 0.31', 'F = 0.36',
       'F = 0.41', 'F = 0.46', 'S = 0.01', 'S = 0.06', 'S = 0.11',
       'S = 0.16', 'S = 0.21', 'S = 0.26', 'S = 0.31', 'S = 0.36',
       'S = 0.41', 'S = 0.46', 'Nominal'], dtype=object)
```

مشاهده می‌شود که تمامی داده‌ها در این دو گروه، از جنس عدد نیستند؛ درنتیجه نیازمند جایگزینی با اعداد مناسب هستند. برای هر داده در گروه تفصیلی می‌توان عدد مقابل آن‌را به عنوان یک ویژگی برای اعلام اهمیت داده استفاده کرد. برای گروه کلاس هم هر کلاس را با کمک دیکشنری به ۰، ۱، ۲، ۳ و ۴ اختصاص داد. کد مربوطه به صورت زیر می‌باشد.

```
replacements = {'Nominal': 0, 'Lean': 1, 'ExcessAir': 2, 'Fouling': 3,'Scaling' : 4,'%=0.05' : 0.05, '%=0.1': 0.1, '%=0.15': 0.15,
                '%=0.20': 0.2, '%=0.25' : 0.25, '%=0.3': 0.3, '%=0.35': 0.35, '%=0.40': 0.4,  '%=0.45': 0.45, '%=0.50': 0.5, 'F = 0.01': 0.01,
                'F = 0.06': 0.06, 'F = 0.11': 0.11, 'F = 0.16': 0.16, 'F = 0.21': 0.21, 'F = 0.26': 0.26, 'F = 0.31': 0.31, 'F = 0.36': 0.36,
                'F = 0.41': 0.41, 'F = 0.46': 0.46, 'S = 0.01': 0.01,'S = 0.06': 0.06, 'S = 0.11': 0.11, 'S = 0.16': 0.16, 'S = 0.21': 0.21,
                'S = 0.26': 0.26, 'S = 0.31': 0.31, 'S = 0.36': 0.36,'S = 0.41': 0.41, 'S = 0.46': 0.46 }

data=data.replace(replacements)
data.to_csv('data.csv', index=False)
```

داده جدید به صورت زیر تبدیل می‌شود.

| Fuel_Mdot | Tair | Treturn | Tsupply | Water_Mdot | Condition | Class |
|---|---|---|---|---|---|---|
| 1 | 283 | 333 | 363.574744 | 3 | 0.05 | 1 |
| 1 | 283 | 333 | 362.3495168 | 3 | 0.1 | 0 |
| 1 | 283 | 333 | 361.216941 | 3 | 0.15 | 2 |
| 1 | 283 | 333 | 360.1668904 | 3 | 0.2 | 2 |
| 1 | 283 | 333 | 359.1906622 | 3 | 0.25 | 2 |

از مهم‌ترین مرحله‌ها در هر بررسی داده، کاهش بعد برای افزایش سرعت محاسبات است. البته ممکن است موجب از دست رفتن برخی از داده‌های مفید شود اما کمک زیادی در طبقه بندی داده‌ها می‌کند. داده‌ی مورد استفاده در این پروژه دارای ۶ ویژگی قابل آموزش برای شبکه است که می‌توانند کاهش یابند. در ادامه چند روش مهم و پرکاربرد برای کاهش بعد را روی این داده بررسی می‌کنیم:

## روش t-SNE

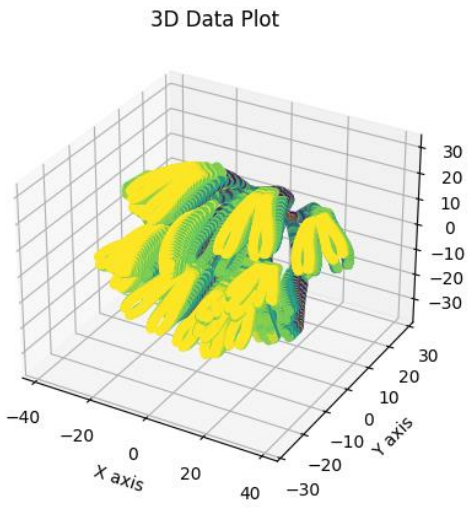
روش t-SNE (t-Distributed Stochastic Neighbor Embedding) یک روش محبوب کاهش بعد غیرخطی است که برای نمایش داده‌های فضاهای بُعدی بالا در یک فضای کمتر بعد استفاده می‌شود. این تکنیک به ویژه برای آشکارسازی خوشه‌ها و الگوها در داده‌ها بسیار مؤثر است.

t-SNE در سال ۲۰۰۸ توسط لورنس فان دِر ماتِن (Laurens van der Maaten) و جفری هینتون (Geoffrey Hinton) معرفی شد. این تکنیک بر پایه مفهوم Stochastic Neighbor Embedding (SNE) استوار است اما برخی از محدودیت‌های آن را برطرف می‌کند. t-SNE یک توزیع احتمالی بر روی جفت‌های اشیاء بُعد بالا و یک توزیع مشابه بر روی جفت‌های متناظر آن‌ها در بُعد کمتر بنا می‌کند. هدف آن کاهش اختلاف بین این دو توزیع را با بهینه‌سازی تعبیه‌گر است.

کد استفاده شده به صورت زیر می‌باشد.

```
from matplotlib import pyplot as plt
from sklearn.manifold import TSNE

t=TSNE(n_components=3,random_state=42)
X_embed=t.fit_transform(X)
```

خروجی داده بدست آمده به صورت زیر است.




## روش Autoencoder

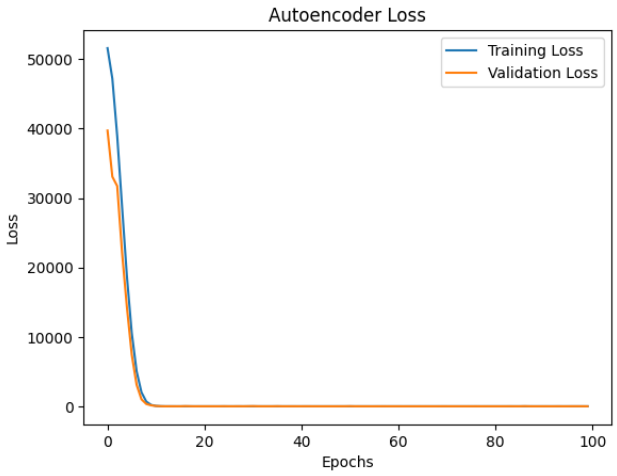
اتوانکدر (Autoencoder) نوعی شبکه عصبی مصنوعی است که برای یادگیری بدون ناظر و کاهش بُعد استفاده می‌شود. این شبکه شامل یک شبکه کدگذار است که داده ورودی را به یک نمایش کم‌بُعدی، که "کُدگذاری" نامیده می‌شود، نگاشت می‌دهد و یک شبکه بازگشتی است که سعی در بازسازی داده ورودی از کُدگذاری دارد. هدف اتوانکدر کمینه کردن خطای بازسازی است که مدل را به یادگیری نماینده‌های معنادار از داده ورودی تشویق می‌کند.

مفهوم اتوانکدر در دهه ۱۹۸۰ میلادی توسط جفری هینتون و همکارانش معرفی شد. با امکاناتی که با پیشرفت تکنیک‌های یادگیری عمیق، به خصوص با ظهور اتوانکدرهای عمیق و اتوانکدرهای واریانسی، به دست آمد، محبوبیت اتوانکدرها بیشتر شد.

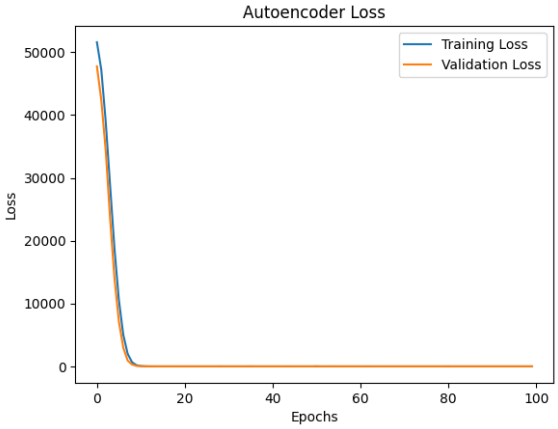
برای بررسی بیشتر این روش روی داده‌ها، از دو حالت ۳ ویژگی و ۴ ویژگی در خروجی به صورت جداگانه خروجی گرفته شده‌است. کد این روش با استفاده از کتابخانه keras به صورت زیر می‌باشد.

```python
# Define the autoencoder model
input_layer = Input(shape=(input_dim,))
encoder = Dense(128, activation='relu')(input_layer)
encoder = BatchNormalization()(encoder)
encoder = Dense(32, activation='relu')(encoder)
encoder = BatchNormalization()(encoder)
encoder = Dense(16, activation='relu')(encoder)
encoder = BatchNormalization()(encoder)
encoder = Dense(8, activation='relu')(encoder)
encoder = BatchNormalization()(encoder)
encoder = Dense(encoding_dim, activation='relu')(encoder)

decoder = Dense(8, activation='relu')(encoder)
decoder = BatchNormalization()(decoder)
decoder = Dense(16, activation='relu')(encoder)
decoder = BatchNormalization()(decoder)
decoder = Dense(32, activation='relu')(encoder)
decoder = BatchNormalization()(decoder)
decoder = Dense(128, activation='relu')(decoder)
decoder = BatchNormalization()(decoder)
decoder = Dense(input_dim, activation='linear')(decoder)

# Create the autoencoder model
autoencoder = Model(inputs=input_layer, outputs=decoder)

import keras
opt = keras.optimizers.Adam(learning_rate=0.001)
# Compile the model
autoencoder.compile(optimizer=opt, loss='mean_squared_error')

# Train the autoencoder
history = autoencoder.fit(X_train, X_train, epochs=100, batch_size=128, validation_data=(X_test, X_test))
```

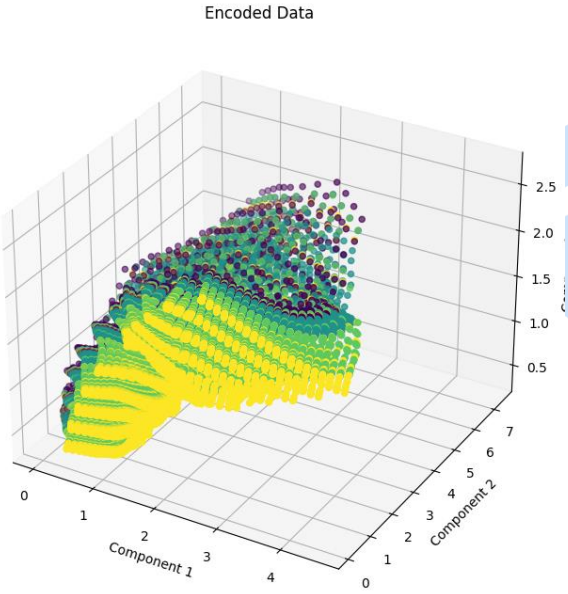
نمودار loss برای ۳ ویژگی به صورت زیر بدست آمده است.



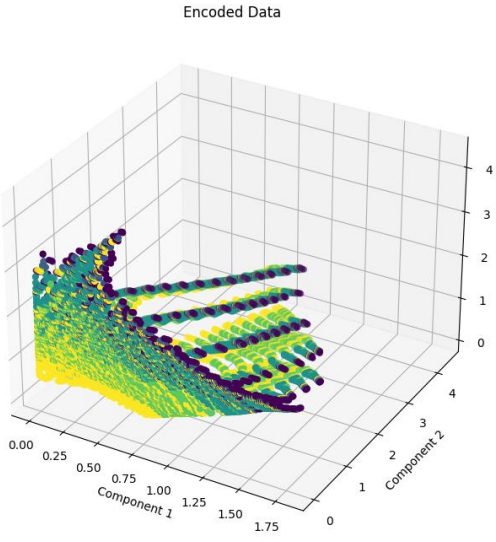

برای ۴ ویژگی نیز به صورت مشابه بدست آمده است.




نمایش داده‌های انکد شده برای ۳ ویژگی به صورت زیر می‌باشد.




برای ۴ ویژگی نیز به صورت زیر بدست آمده که در اینجا ۳ ویژگی اول نمایش داده شده‌است.



## روش PCA

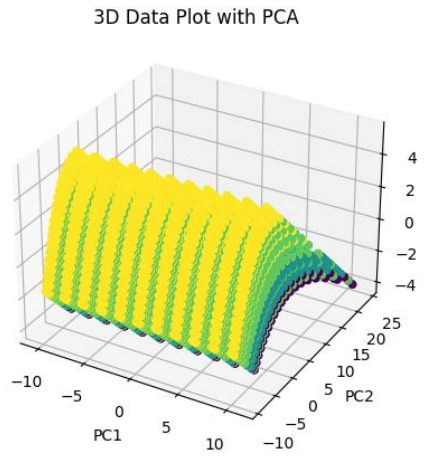
PCA (تحلیل مؤلفه‌های اصلی) یکی از محبوب‌ترین تکنیک‌های کاهش بُعد خطی است که برای تبدیل داده‌های فضاهای بُعدی بالا به یک فضای بُعد کمتر با حفظ اطلاعات مهم استفاده می‌شود. این تکنیک با شناسایی مؤلفه‌های اصلی، که جهت‌های عمودی‌ای هستند و بیشترین واریانس را در داده‌ها ثبت می‌کنند، این کاهش بُعد را انجام می‌دهد.

PCA در سال ۱۹۰۱ توسط کارل پیرسون معرفی شد، اما فرمولاسیون مدرن آن و استفاده گسترده آن مربوط به پژوهشگران مختلفی از جمله هارولد هاتلینگ و هربرت آ. دیوید است. از آن زمان به بعد، PCA به عنوان یک تکنیک بنیادی در زمینه‌های مختلف از جمله تحلیل داده‌ها، تشخیص الگو و یادگیری ماشین شناخته شده است.

کد این روش برای ۳ ویژگی در خروجی به شرح زیر می‌باشد.

```
# Apply PCA
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X)
```

خروجی بدست آمده از این روش به صورت زیر می‌باشد.




## روش LDA

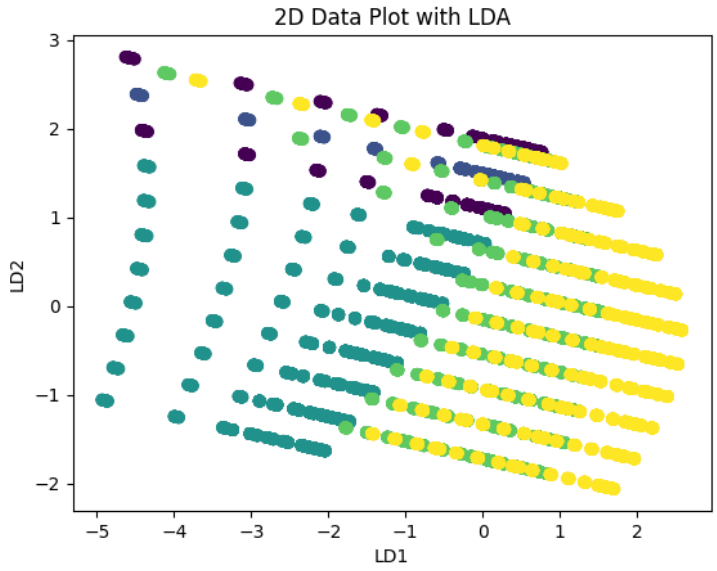
LDA (تحلیل ممیز خطی) یک تکنیک کاهش بعد است که هدف آن پیدا کردن ترکیب خطی از ویژگی‌ها است که بیشینه جدایی بین کلاس‌ها در داده را فراهم می‌کند. این تکنیک به طور معمول در وظایف طبقه‌بندی استفاده می‌شود تا داده‌ها را روی یک فضای کم‌بعدی نگاشت کند و هم‌گسیلی کلاس‌ها را بیشینه سازد.

LDA در سال ۱۹۳۶ توسط رونالد آ. فیشر معرفی شد به عنوان یک روش برای یافتن یک ممیز خطی که دو کلاس را به حداکثر جدا می‌کند. در طول سال‌ها، LDA بهبود و گسترش یافته و برای حل مسائل چندکلاسه نیز به‌کار گرفته می‌شود.

کد این روش به صورت زیر می‌باشد. به دلیل محدودیت‌هایی که داده نسبت به این الگوریتم داشت، تنها ۲ ویژگی قابل استخراج بود.

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Apply LDA
lda = LinearDiscriminantAnalysis(n_components=2)
X_lda = lda.fit_transform(X, y)
```

خروجی این روش به صورت زیر است.




تنها موردی که باعث نگرانی است این است که داده‌های زیادی ممکن است در کاهش بعد از ۶ ویژگی به ۲ ویژگی رخ دهد.

## مقایسه روش‌های مختلف

برای بررسی بهتر داده های بدست آمده از هر روش معیارهای Silhouette Score، Calinski-Harabasz Index و Davies-Bouldin Index استفاده می‌کنیم. تحلیل هر معیار به صورت زیر می‌باشد.

معیار Silhouette Score اندازه‌گیری می‌کند که هر نمونه در یک مجموعه داده در مقایسه با خوشه‌های دیگر چقدر در خوشه اختصاص داده شده خود قرار می‌گیرد. از ۱- تا ۱ متغیر است، جایی که

مقدار بالاتر نشان دهنده عملکرد بهتر خوشه بندی است. نمره نزدیک به ۱ نشان می‌دهد که نمونه‌ها به خوبی خوشه‌بندی شده‌اند، در حالی که نمرات نزدیک به ۰ نشان‌دهنده همپوشانی خوشه‌ها و نمرات منفی نشان می‌دهد که ممکن است نمونه‌ها به خوشه اشتباهی اختصاص داده شده باشند. امتیاز Silhouette را می توان برای ارزیابی فشرده بودن و جداسازی خوشه ها، ارائه بینشی در مورد کیفیت بازسازی داده ها یا نتایج خوشه بندی استفاده کرد.

شاخص Calinski-Harabasz که به عنوان معیار نسبت واریانس نیز شناخته می‌شود، اندازه‌گیری نسبت بین پراکندگی درون خوشه‌ای و پراکندگی بین خوشه‌ای است. هدف آن به حداکثر رساندن نسبت است که نشان دهنده خوشه‌های به خوبی جدا شده و فشرده است. مقدار بالاتر Calinski-Harabasz Index مربوط به عملکرد خوشه بندی بهتر است. از این شاخص می توان برای ارزیابی کیفیت تخصیص خوشه ها و فشرده بودن داده های بازسازی شده استفاده کرد.

شاخص Davies-Bouldin شباهت بین خوشه ها را با در نظر گرفتن پراکندگی درون خوشه ای و جدایی بین خوشه ای ارزیابی می کند. میانگین شباهت بین هر خوشه و مشابه ترین خوشه را با در نظر گرفتن اندازه آنها اندازه گیری می کند. مقدار پایین‌تر Davies-Bouldin Index نشان‌دهنده عملکرد بهتر خوشه‌بندی است، با مقادیر کوچک‌تر نشان‌دهنده خوشه‌های فشرده‌تر و متمایزتر است. این شاخص می تواند به ارزیابی کیفیت نتایج خوشه بندی یا بازسازی داده ها با در نظر گرفتن تعادل بین فاصله های درون خوشه ای و بین خوشه ای کمک کند.

مقادیر معیارهای توضیح داده شده در بالا به صورت زیر می‌باشد.

```
Silhouette Score of main data: -0.025457842383237995
Silhouette Score of tsne output data: -0.044015173
Silhouette Score of 3 features AE output data: -0.026696546
Silhouette Score of 4 features AE output data: -0.017811231
Silhouette Score of pca output data: -0.025655223149941713
Silhouette Score of lda output data: 0.05253760398332844

Calinski-Harabasz Index of main data: 647.5649101758748
Calinski-Harabasz Index of tsne output data: 268.0483930135195
Calinski-Harabasz Index of 3 features AE output data: 675.0844408350973
Calinski-Harabasz Index of 4 features AE output data: 411.8250949577895
Calinski-Harabasz Index of pca output data: 657.9766426312244
Calinski-Harabasz Index of lda output data: 5271.187204452654

Davies-Bouldin Index of main data: 242.7819594743384
Davies-Bouldin Index of tsne output data: 254.8824327490932
Davies-Bouldin Index of 3 features AE output data: 177.54053137789325
Davies-Bouldin Index of 4 features AE output data: 293.8286470299288
Davies-Bouldin Index of pca output data: 240.34603143097874
Davies-Bouldin Index of lda output data: 69.30243641901401
```

مشاهده می‌شود که به دلیل پیچیدگی بالای داده، معیار اول خود داده کمتر از صفر شده است. با اینحال، مقدار معیار دوم داده کم نیست و نشان‌گر وجود خوشه‌هایی قابل تمایز از دید واریانسی در داده است. البته در معیار سوم داده اصلی کمی ضعف نسبت به تمایز خوشه‌ها از دید میانگین شباهت بین داده‌ایست.

در کل روش LDA از بین سایر روش‌ها بهتر عمل کرده است. روش دومی که می‌توان آن‌را برگزید نیز روش اتوانکدر با ۳ ویژگی در خروجی است که بعد از LDA عملکرد مناسب تری را نشان می‌دهد.

در ادامه این پروژه سعی می‌شود تا طبقه بندی کننده‌ی مناسبی برای این داده انتخاب شود که فعلا اولین گزینه MLP خواهد بود اما سعی می‌شود تا روش‌هایی مثل RNN نیز بررسی شوند.

# مراجع


https://blog.keras.io/building-autoencoders-in-keras.html

https://ieee-dataport.org/open-access/simulated-boiler-data-fault-detection-and-classification