Nama : Akmal Zuhdy Prasetya
NIM : H071191035
Kelas : Pengantar Data Mining

# Resume Pertemuan 5

## A. Apa itu Dimensionality Reduction?

Dalam masalah klasifikasi machine learning, seringkali ada terlalu banyak faktor yang mendasari klasifikasi akhir dilakukan. Faktor-faktor ini pada dasarnya adalah variabel yang disebut fitur. Semakin tinggi jumlah fitur, semakin sulit untuk memvisualisasikan dataset pelatihan dan kemudian mengerjakannya. Terkadang, sebagian besar fitur ini berkorelasi, dan karenanya fitur tersebut menjadi berlebihan. Di sinilah algoritma pengurangan dimensi berperan. Pengurangan dimensi adalah proses mengurangi jumlah variabel acak yang dipertimbangkan, dengan memperoleh satu set variabel utama. Ini dapat dibagi menjadi seleksi fitur dan ekstraksi fitur.

## B. Mengapa Dimensionality Reduction?

Kutukan dimensi adalah kondisi yang terjadi ketika kita ingin mengklasifikasikan, mengatur, dan menganalisis data berdimensi tinggi. Ketika jumlah dimensi meningkat, jarak antara dua titik independen meningkat, dan kesamaan berkurang. Masalah ini menghasilkan lebih banyak kesalahan dalam hasil akhir kami setelah penambangan data. Ketika kita mengerjakan data, terutama data besar, maka jumlah titik data yang sangat besar, jadi banyak dimensi yang mungkin ada. Dalam hal ini, hampir tidak mungkin untuk mendapatkan hasil yang diinginkan dan bahkan jika kita mengira itu mungkin maka itu akan memberikan hasil yang tidak efisien.

Berikut beberapa alasan mengapa pengurangan dimensi penting dilakukan:

- Jika kita mengurangi dimensi, maka akan lebih mudah dan nyaman untuk mengumpulkan data.
- Data tidak dikumpulkan hanya untuk data mining.
- Data terakumulasi dengan kecepatan yang baik.
- Pra-pemrosesan data adalah tugas penting yang harus dilakukan untuk penambangan data yang lebih baik dan efektif.
- Pengurangan dimensi adalah pendekatan yang efektif untuk mengumpulkan lebih sedikit data tetapi data yang efisien.
- Pengurangan Dimensi sangat membantu dalam proyeksi data berdimensi tinggi ke dalam Visualisasi 2D atau 3D.
- Pengurangan Dimensi sangat membantu dalam penyimpanan dan pengambilan data yang tidak efisien dan mempromosikan konsep kompresi Data.
- Pengurangan Dimensi mendorong efek positif pada akurasi kueri dengan penghapusan Noise.

- Pengurangan Dimensi mengurangi waktu komputasi. Ini mempercepat waktu yang dibutuhkan untuk melakukan perhitungan yang sama.
- Pengurangan Dimensi sangat membantu untuk menghapus fitur yang berlebihan.

Beberapa aplikasi dari pengurangan dimensi adalah sebagai berikut:

- Text Mining (Penambangan Teks)
- Image Retrieval (Pengambilan Gambar)
- Microarray Data Analysis (Analisis Data Microarray)
- Protein Classification (Klasifikasi Protein)
- Face & Image Recognition (Pengenalan Wajah & Gambar)
- Intrusion Detection (Deteksi Gangguan)
- Customer Relationship Management (Pengelolaan Hubungan Pelanggan)
- Handwritten Digit Recognition (Pengenalan Digit Tulisan Tangan)

C. Feature Selection (Seleksi Fitur)

Secara sederhana, seleksi fitur adalah tentang memilih subset fitur dari fitur asli untuk mengurangi kompleksitas model, meningkatkan efisiensi komputasi model dan mengurangi kesalahan generalisasi yang disebabkan oleh noise oleh fitur yang tidak relevan. Berikut ini merupakan beberapa teknik pemilihan fitur yang penting:

- Regularization Techniques atau Teknik Regularisasi seperti regularisasi norma $L_1$ yang menghasilkan sebagian besar bobot fitur menjadi nol.
- Feature Importance Techniques atau Teknik Kepentingan Fitur seperti menggunakan estimator seperti algoritma Random Forest agar sesuai dengan model dan memilih fitur berdasarkan nilai atribut seperti feature_importances_.
- Greedy Search Algorithm atau Algoritma Pencarian Serakah seperti beberapa berikut ini yang berguna untuk algoritma (seperti K-nearest neighbor, K-NN) di mana teknik regularisasi tidak didukung.

Menurut data pelatihan yang digunakan (berlabel, tidak berlabel, atau berlabel sebagian), metode seleksi fitur dapat dibagi menjadi model terawasi, tidak terawasi, dan semi terawasi. Menurut hubungannya dengan metode pembelajaran, metode seleksi fitur dapat diklasifikasikan menjadi berikut:

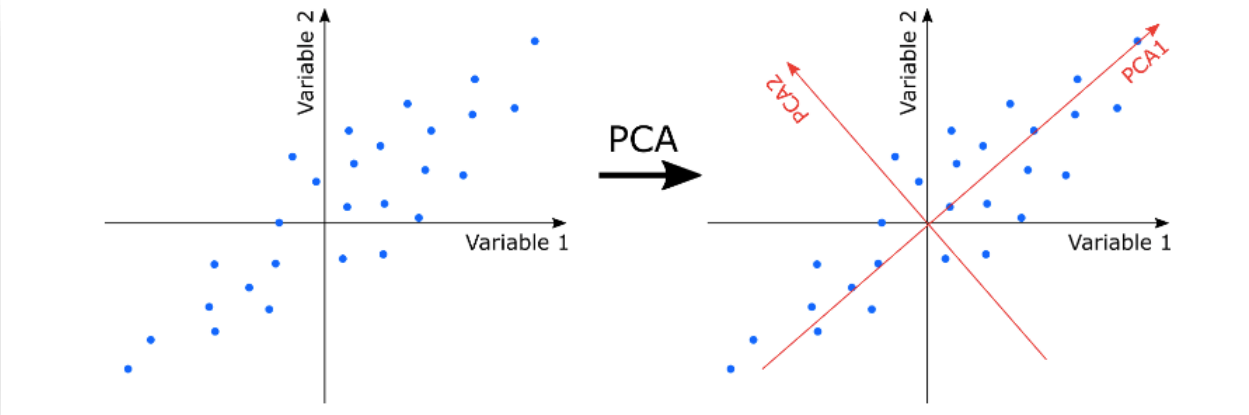
- Filter Method, model filter hanya mempertimbangkan hubungan antara fitur dan label kelas.
- Wrapper Method, mengukur “kegunaan” fitur berdasarkan performa classifier.
- Embedded Method, dalam metode tertanam, fitur dipilih dalam proses pelatihan model pembelajaran, dan hasil pemilihan fitur keluar secara otomatis saat proses pelatihan selesai. Melatih model regresi Lasso adalah contoh klasik metode tersemat untuk pemilihan fitur.

D. Feature Extraction (Ekstraksi Fitur)

Ekstraksi fitur adalah tentang mengekstraksi/mengambil informasi dari kumpulan fitur asli untuk membuat subruang fitur baru. Ide utama di balik ekstraksi fitur adalah untuk mengompresi data dengan tujuan mempertahankan sebagian besar informasi yang relevan. Seperti teknik pemilihan fitur, teknik ini juga digunakan untuk mengurangi jumlah fitur dari set fitur asli untuk mengurangi kompleksitas model, model overfitting, meningkatkan efisiensi komputasi model dan mengurangi kesalahan generalisasi. Berikut ini adalah macam-macam teknik ekstraksi fitur:

1. PCA (Principal Component Analysis)

Analisis komponen utama adalah teknik transformasi linier tanpa pengawasan (unsupervised) yang utamanya digunakan untuk ekstraksi fitur dan pengurangan dimensi. Ini bertujuan untuk menemukan arah varians maksimum dalam data berdimensi tinggi dan memproyeksikan data ke subruang baru dengan dimensi yang sama atau lebih sedikit dari yang asli. Dalam diagram yang diberikan di bawah ini, perhatikan arah varians maksimum data. Ini diwakili menggunakan PCA1 (varian maksimum pertama) dan PC2 (varian maksimum kedua).




Berikut ini merupakan tahapan yang harus dilalui untuk menerapkan PCA:

- Lakukan One-Hot Encoding untuk mengubah kumpulan data kategorikal menjadi kumpulan data numerik
- Lakukan pelatihan / uji pemisahan dataset
- Standarisasi set data pelatihan dan pengujian
- Bangun matriks kovarians dari kumpulan data pelatihan
- Membangun eigendecomposition dari matriks kovarians
- Pilih fitur yang paling penting menggunakan varians yang dijelaskan
- Bangun matriks proyek
- Ubah kumpulan data pelatihan menjadi subruang fitur baru

2. LDA (Linear Discriminant Analysis)

Analisis diskriminan linier, juga disebut analisis diskriminan normal (NDA) atau analisis fungsi diskriminan adalah generalisasi dari diskriminan linier Fisher, metode yang digunakan dalam statistik dan bidang lainnya, untuk menemukan kombinasi linier fitur yang menjadi ciri atau memisahkan dua atau lebih kelas dari objek atau peristiwa. Kombinasi yang dihasilkan dapat digunakan sebagai pengklasifikasi linier, atau, lebih umum, untuk pengurangan dimensi sebelum klasifikasi selanjutnya.

LDA terkait erat dengan analisis komponen utama (PCA) dan analisis faktor karena keduanya mencari kombinasi linear variabel yang paling baik menjelaskan data. LDA secara eksplisit mencoba untuk memodelkan perbedaan antara kelas data. PCA, sebaliknya, tidak memperhitungkan perbedaan kelas, dan analisis faktor membangun kombinasi fitur berdasarkan perbedaan daripada persamaan. Analisis diskriminan juga berbeda dari analisis faktor dalam hal ini bukan teknik saling ketergantungan: perbedaan antara variabel bebas dan variabel terikat (juga disebut variabel kriteria) harus dibuat.

Berikut ini merupakan tahapan yang harus dilalui untuk menerapkan LDA:

- Menghitung vektor rata-rata dari setiap kelas variabel terikat
- Komputer dengan matriks pencar dalam kelas dan antar kelas
- Menghitung nilai eigen dan vektor eigen untuk SW(Matriks sebar di dalam kelas) dan SB (matriks sebar antar kelas)
- Urutkan nilai eigen dalam urutan menurun dan pilih k . teratas
- Membuat matriks baru yang berisi vektor eigen yang dipetakan ke nilai k eigen
- Memperoleh fitur baru (yaitu diskriminan linier) dengan mengambil produk titik dari data dan matriks.

3. t-SNE (Distributed Stochastic Neighbor Embedding)

Pada dasarnya t-SNE adalah teknik pengurangan dimensi, sebanding dengan analisis komponen utama atau PCA. Oleh karena itu, ini juga dapat dianggap sebagai alternatif rekayasa fitur. Namun, PCA adalah teknik pengurangan dimensi linier yang berupaya memaksimalkan varians dan mempertahankan jarak berpasangan yang besar. Dengan kata lain, hal-hal yang berbeda berakhir berjauhan. Hal ini dapat menyebabkan pengurangan suboptimal dengan data non-linear. Sebaliknya, t-SNE berusaha mempertahankan kesamaan lokal dengan berfokus pada jarak berpasangan yang kecil.

Teknik reduksi dimensi t-SNE bekerja dalam 2 langkah. Pada langkah 1, distribusi probabilitas yang mewakili ukuran kesamaan atas pasangan titik data berdimensi tinggi dibangun. Pada Langkah 2, distribusi probabilitas serupa atas titik data di peta dimensi rendah dibangun. Divergensi Kullback–Leibler (juga dikenal sebagai perolehan informasi atau entropi

relatif) antara 2 distribusi kemudian diminimalkan. Ingatlah bahwa divergensi Kullback–Leibler adalah ukuran kesamaan antara dua distribusi probabilitas.

## E. Implementasi Algoritma

Berikut adalah implementasi dari Dimensionality Reduction dengan menggunakan dataset tentang tipe tipe bintang. Untuk detail lebih lengkap dari dataset ini bisa dilihat pada link berikut: https://www.kaggle.com/brsdincer/star-type-classification.

### 1. Import Library & Dataset

Meng-import library untuk tahapan awal pengambilan dataset.

```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
% matplotlib inline

os.environ['KAGGLE_USERNAME'] = "akmalzuhdyprasetya"
os.environ['KAGGLE_KEY'] = "3f1fd1ffa5d36294293dfb098b3b7392"

!kaggle datasets download -d brsdincer/star-type-classification

!unzip -q /content/star-type-classification.zip
```

Menampilkan dataset.

```
data = pd.read_csv("/content/Stars.csv")
data
```

| | Temperature | L | R | A_M | Color | Spectral_Class | Type |
|---|---|---|---|---|---|---|---|
| 0 | 3068 | 0.002400 | 0.1700 | 16.12 | Red | M | 0 |
| 1 | 3042 | 0.000500 | 0.1542 | 16.60 | Red | M | 0 |
| 2 | 2600 | 0.000300 | 0.1020 | 18.70 | Red | M | 0 |
| 3 | 2800 | 0.000200 | 0.1600 | 16.65 | Red | M | 0 |
| 4 | 1939 | 0.000138 | 0.1030 | 20.06 | Red | M | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 235 | 38940 | 374830.000000 | 1356.0000 | -9.93 | Blue | O | 5 |
| 236 | 30839 | 834042.000000 | 1194.0000 | -10.63 | Blue | O | 5 |
| 237 | 8829 | 537493.000000 | 1423.0000 | -10.73 | White | A | 5 |
| 238 | 9235 | 404940.000000 | 1112.0000 | -11.23 | White | A | 5 |
| 239 | 37882 | 294903.000000 | 1783.0000 | -7.80 | Blue | O | 5 |

240 rows × 7 columns

2. Pre-Processing Data
   - Menampilkan jumlah tipe dari kelas target.

```
[223] data["Type"].value_counts()

      5    40
      4    40
      3    40
      2    40
      1    40
      0    40
      Name: Type, dtype: int64
```

   - Melakukan Label Encoding terhadap kolom Color dan Spectral_Class karena bertipe object.

```
Temperature         int64
L                 float64
R                 float64
A_M               float64
Color              object
Spectral_Class     object
Type              float64
dtype: object
```

```
# label encoding
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
data['Color'] = le.fit_transform(data['Color'])
data['Spectral_Class'] = le.fit_transform(data['Spectral_Class'])
```

   - Mengeluarkan kolom target Type dari tabel.

```
x = data.drop("Type", 1)
y = data["Type"]
column_name = list(x.columns)
print(x)
```

```
     Temperature             L          R    A_M  Color  Spectral_Class
0           3068      0.002400     0.1700  16.12      8               5
1           3042      0.000500     0.1542  16.60      8               5
2           2600      0.000300     0.1020  18.70      8               5
3           2800      0.000200     0.1600  16.65      8               5
4           1939      0.000138     0.1030  20.06      8               5
..           ...           ...        ...    ...    ...             ...
235        38940  374830.000000  1356.0000  -9.93      0               6
236        30839  834042.000000  1194.0000 -10.63      0               6
237         8829  537493.000000  1423.0000 -10.73      9               0
238         9235  404940.000000  1112.0000 -11.23      9               0
239        37882  294903.000000  1783.0000  -7.80      0               6

[240 rows x 6 columns]
```

- Melihat statistik dataset.

```
data.describe()
```

| | Temperature | L | R | A_M | Color | Spectral_Class | Type |
|---|---|---|---|---|---|---|---|
| count | 240.000000 | 240.000000 | 240.000000 | 240.000000 | 240.000000 | 240.000000 | 240.000000 |
| mean | 10497.462500 | 107188.361635 | 237.157781 | 4.382396 | 5.766667 | 3.758333 | 2.500000 |
| std | 9552.425037 | 179432.244940 | 517.155763 | 10.532512 | 4.208446 | 2.090007 | 1.711394 |
| min | 1939.000000 | 0.000080 | 0.008400 | -11.920000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 3344.250000 | 0.000865 | 0.102750 | -6.232500 | 1.000000 | 1.000000 | 1.000000 |
| 50% | 5776.000000 | 0.070500 | 0.762500 | 8.313000 | 8.000000 | 5.000000 | 2.500000 |
| 75% | 15055.500000 | 198050.000000 | 42.750000 | 13.697500 | 8.000000 | 5.000000 | 4.000000 |
| max | 40000.000000 | 849420.000000 | 1948.500000 | 20.060000 | 16.000000 | 6.000000 | 5.000000 |

- Melakukan normalisasi data agar rentang datanya menjadi lebih kecil.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(x)

print('Min : ', data_scaled.min(axis=0))
print('Max : ', data_scaled.max(axis=0))
```

```
Min :  [0. 0. 0. 0. 0. 0.]
Max :  [1. 1. 1. 1. 1. 1.]
```

- Menampilkan tabel dengan data yang telah dinormalisasikan.

```
data_new = pd.DataFrame(data=data_scaled, columns= column_name)
data_new.head(11)
```

| | Temperature | L | R | A_M | Color | Spectral_Class |
|---|---|---|---|---|---|---|
| 0 | 0.029663 | 2.731275e-09 | 0.000083 | 0.876798 | 0.5 | 0.833333 |
| 1 | 0.028980 | 4.944550e-10 | 0.000075 | 0.891807 | 0.5 | 0.833333 |
| 2 | 0.017367 | 2.590003e-10 | 0.000048 | 0.957473 | 0.5 | 0.833333 |
| 3 | 0.022622 | 1.412729e-10 | 0.000078 | 0.893371 | 0.5 | 0.833333 |
| 4 | 0.000000 | 6.828189e-11 | 0.000049 | 1.000000 | 0.5 | 0.833333 |
| 5 | 0.023673 | 6.710461e-10 | 0.000052 | 0.903690 | 0.5 | 0.833333 |
| 6 | 0.018339 | 7.652280e-10 | 0.000061 | 0.911194 | 0.5 | 0.833333 |
| 7 | 0.017367 | 3.767276e-10 | 0.000045 | 0.916823 | 0.5 | 0.833333 |
| 8 | 0.018681 | 7.181371e-10 | 0.000052 | 0.918386 | 0.5 | 0.833333 |
| 9 | 0.019994 | 1.177274e-10 | 0.000062 | 0.874609 | 0.5 | 0.833333 |
| 10 | 0.043640 | 3.319912e-09 | 0.000257 | 0.707004 | 0.5 | 0.833333 |

- Menampilkan fitur Color dalam boxplot.

```
import seaborn as sns
sns.boxplot(y = data_new['Color'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2096b562d0>
```

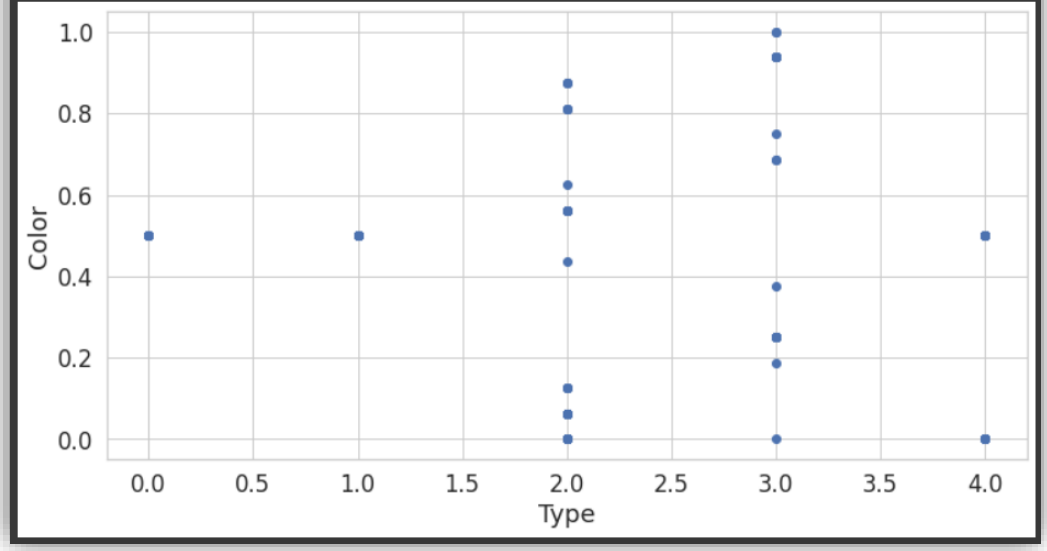

- Melakukan concatenation atau perangkaian terhadap fitur Color yang telah diboxplotkan ke dalam data yang telah dinormalisasikan.

```
data_new = pd.concat([data_new, y], axis = 1)
data_new.head(11)
```

| | Temperature | L | R | A_M | Color | Spectral_Class | Type |
|---|---|---|---|---|---|---|---|
| 0 | 0.029663 | 2.731275e-09 | 0.000083 | 0.876798 | 0.5 | 0.833333 | 0.0 |
| 1 | 0.028980 | 4.944550e-10 | 0.000075 | 0.891807 | 0.5 | 0.833333 | 0.0 |
| 2 | 0.017367 | 2.590003e-10 | 0.000048 | 0.957473 | 0.5 | 0.833333 | 0.0 |
| 3 | 0.022622 | 1.412729e-10 | 0.000078 | 0.893371 | 0.5 | 0.833333 | 0.0 |
| 4 | 0.000000 | 6.828189e-11 | 0.000049 | 1.000000 | 0.5 | 0.833333 | 0.0 |
| 5 | 0.023673 | 6.710461e-10 | 0.000052 | 0.903690 | 0.5 | 0.833333 | 0.0 |
| 6 | 0.018339 | 7.652280e-10 | 0.000061 | 0.911194 | 0.5 | 0.833333 | 0.0 |
| 7 | 0.017367 | 3.767276e-10 | 0.000045 | 0.916823 | 0.5 | 0.833333 | 0.0 |
| 8 | 0.018681 | 7.181371e-10 | 0.000052 | 0.918386 | 0.5 | 0.833333 | 0.0 |
| 9 | 0.019994 | 1.177274e-10 | 0.000062 | 0.874609 | 0.5 | 0.833333 | 0.0 |
| 10 | 0.043640 | 3.319912e-09 | 0.000257 | 0.707004 | 0.5 | 0.833333 | 1.0 |

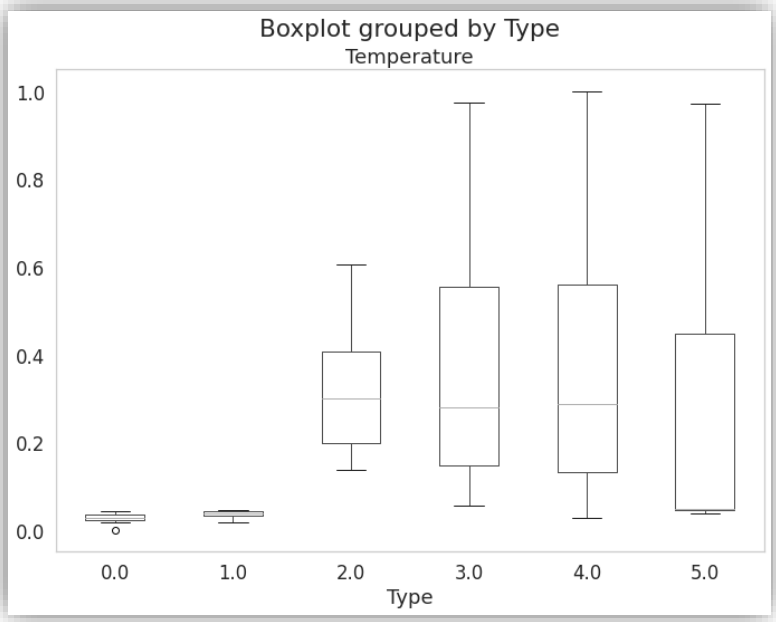
- Menampilkan fitur Color setelah proses perangkaian dengan menggunakan subplot.

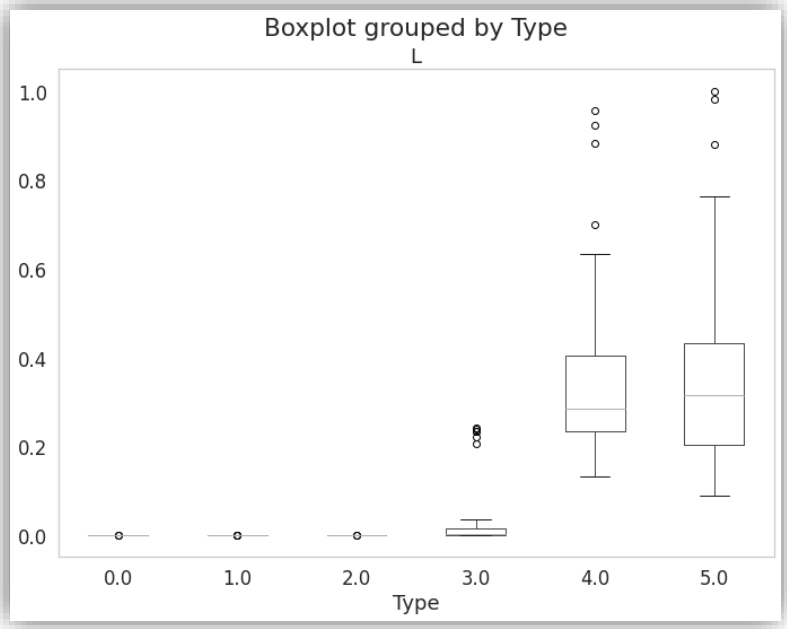
```
fig, ax = plt.subplots(figsize=(12,6))
ax.scatter(data_new['Type'], data_new['Color'])
ax.set_xlabel('Type')
ax.set_ylabel('Color')
plt.show()
```

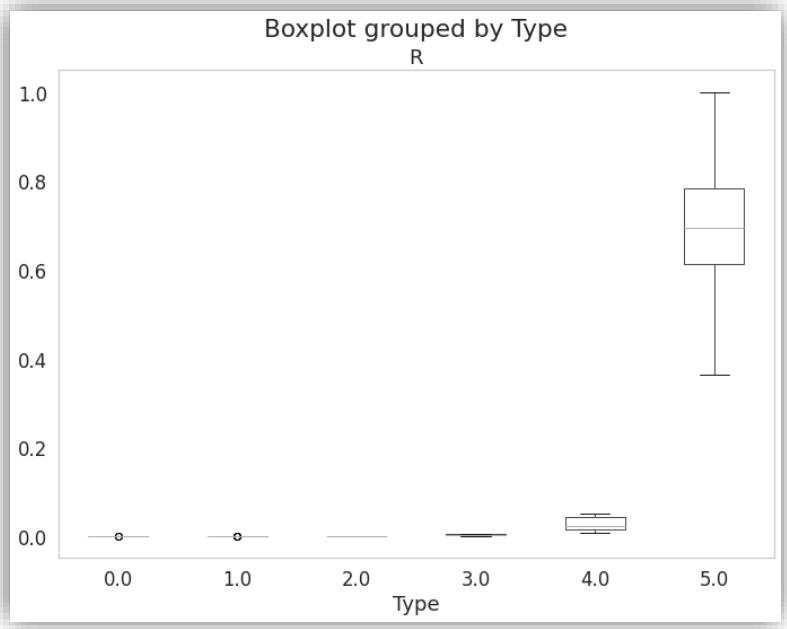


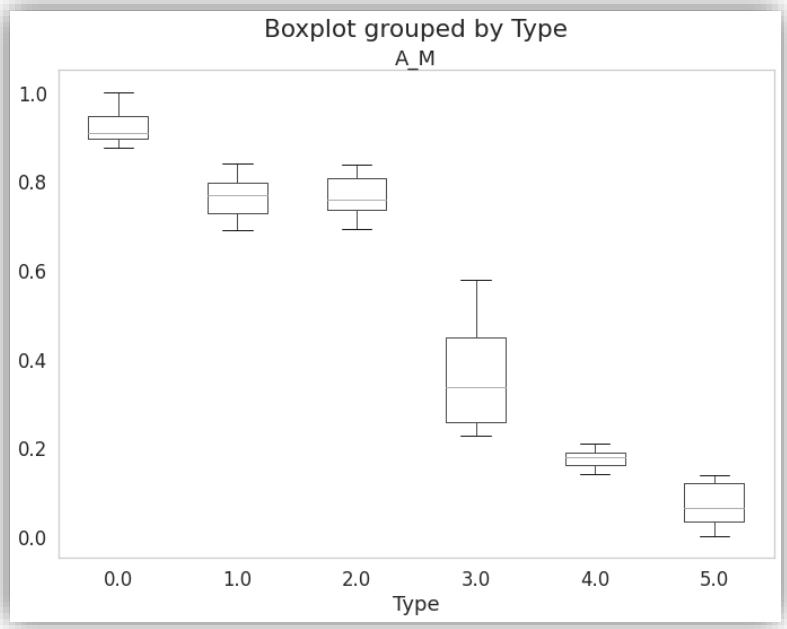

- Menampilkan semua fitur dalam tabel dengan boxplot.

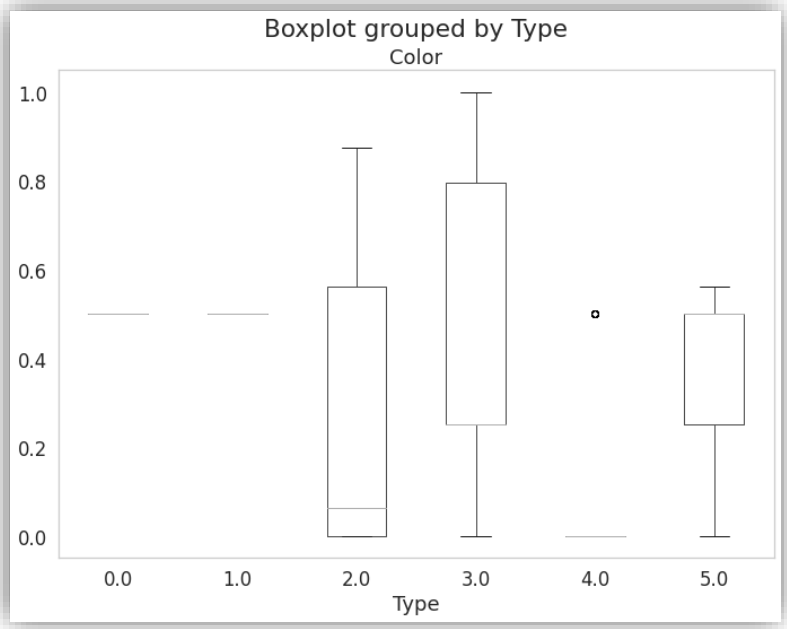
```
sns.set(rc={'figure.figsize':(11, 8)}, font_scale=1.5, style='whitegrid')

x = data_new.drop("Type", 1)
columns_name = list(x.columns)

for i in column_name:
  data_new.boxplot(by ='Type', column =[i], grid = False)
```

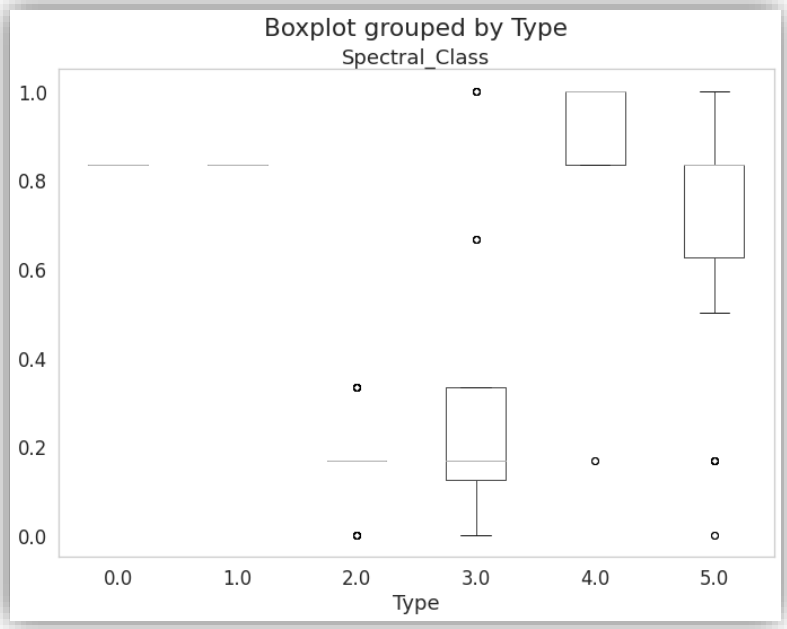

Boxplot grouped by Type
Temperature
1.0
0.8
0.6
0.4
0.2
0.0
0.0
1.0
2.0
3.0
4.0
5.0
Type


Boxplot grouped by Type
L
1.0
0.8
0.6
0.4
0.2
0.0
0.0
1.0
2.0
3.0
4.0
5.0
Type


Boxplot grouped by Type
R
1.0
0.8
0.6
0.4
0.2
0.0
0.0
1.0
2.0
3.0
4.0
5.0
Type


Boxplot grouped by Type
A_M
1.0
0.8
0.6
0.4
0.2
0.0
0.0
1.0
2.0
3.0
4.0
5.0
Type


Boxplot grouped by Type
Color
1.0
0.8
0.6
0.4
0.2
0.0
0.0
1.0
2.0
3.0
4.0
5.0
Type


Boxplot grouped by Type
Spectral_Class
1.0
0.8
0.6
0.4
0.2
0.0
0.0
1.0
2.0
3.0
4.0
5.0
Type

- Melihat Interquantile Range dari data dan shapenya.

```
Q1 = data_new.quantile(0.25)
Q3 = data_new.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
data_new.shape
```

```
Temperature       0.307697
L                 0.233159
R                 0.021887
A_M               0.623202
Color             0.437500
Spectral_Class    0.666667
Type              2.000000
dtype: float64
(240, 7)
```

- Mengubah shape dataset berdasarkan nilai kuartil-kuartil sebelumnya.

```
data_out = data_new[~((data_new < (Q1 - 1.5 * IQR)) |(data_new > (Q3 + 1.5 * IQR))).any(axis=1)]
print(data_out.shape)
```

```
(188, 7)
```

3. Feature Selection
   - Univariate Selelction

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

x = data_out.iloc[:,0:6]  # independent columns
y = data_out.iloc[:,-1]   # target column

bestfeatures = SelectKBest(score_func=chi2, k=6)

fit = bestfeatures.fit(x,y)
dfscores = pd.DataFrame(fit.scores_)
p_values = pd.DataFrame(fit.pvalues_)
dfcolumns = pd.DataFrame(x_fix.columns)

#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores, p_values],axis=1)
featureScores.columns = ['Feature','chi-square Score', 'p-values']  # naming the dataframe columns
print(featureScores.nlargest(6,'chi-square Score'))  # print ranked best features
```

```
          Feature  chi-square Score      p-values
1               L         41.153074  2.498649e-08
5  Spectral_Class         33.936310  7.679401e-07
3             A_M         22.214227  1.816697e-04
0     Temperature         18.229678  1.112845e-03
4           Color         10.445953  3.354968e-02
2               R          3.631822  4.581195e-01
```

Melakukan ranking fitur terbaik dengan bantuan package SelectKBeat dan chi2.

- SVM-RFE

  Membuat method yang akan menyimpan ranking dari fitur ke dalam sebuah dictionary.

```
ranks = {}
# Create our function which stores the feature rankings to the ranks dictionary
def ranking(ranks, names, order=1):
    minmax = MinMaxScaler()
    ranks = minmax.fit_transform(order*np.array([ranks]).T).T[0]
    ranks = map(lambda x: round(x,2), ranks)
    return dict(zip(names, ranks))
```

  Melakukan ranking fitur dengan metode SVM-RFE yang melibatkan beberapa package seperti terlihat pada screenshot kode di bawah.

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import fbeta_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from scipy import interp
from sklearn.metrics import roc_auc_score
from sklearn.feature_selection import RFE
from operator import itemgetter

x = data_out.iloc[:,0:6]
y = data_out.iloc[:,-1]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,
                                                    stratify = y)

colnames = x.columns
n_features_to_select = 1
svclassifier = SVC(kernel='linear')
rfe = RFE(svclassifier, n_features_to_select, verbose = 1)
rfe.fit(x, y)

y_pred = rfe.predict(x)

ranks[f"RFE score"] = ranking(list(map(float, rfe.ranking_)), colnames,
                              order=-1)
```

```
Fitting estimator with 6 features.
Fitting estimator with 5 features.
Fitting estimator with 4 features.
Fitting estimator with 3 features.
Fitting estimator with 2 features.
```

Mengurutkan dan menampilkan hasil ranking fitur.

```
# Put the mean scores into a Pandas dataframe
data_rank = pd.DataFrame(list(ranks[f"RFE score"].items()), columns= ['Feature','Scores'])

# Sort the dataframe
data_rank = data_rank.sort_values('Scores', ascending=False)
```

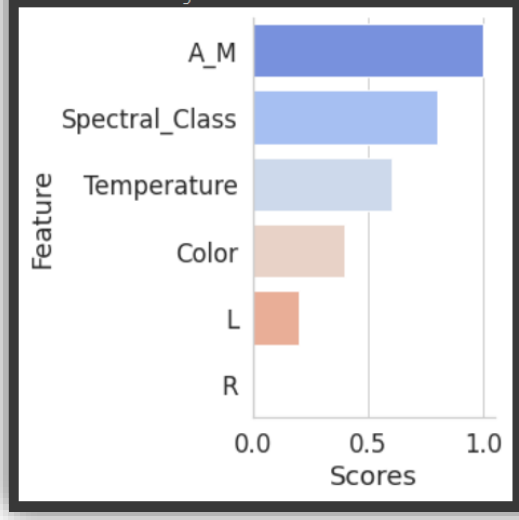
```
data_rank
```

| | Feature | Scores |
|---|---|---|
| 3 | A_M | 1.0 |
| 5 | Spectral_Class | 0.8 |
| 0 | Temperature | 0.6 |
| 4 | Color | 0.4 |
| 1 | L | 0.2 |
| 2 | R | 0.0 |

Menampilkan hasil ranking dengan factorplot.

```
sns.factorplot(x="Scores", y="Feature", data = data_rank, kind="bar", size=5,
               aspect=1, palette='coolwarm')
```

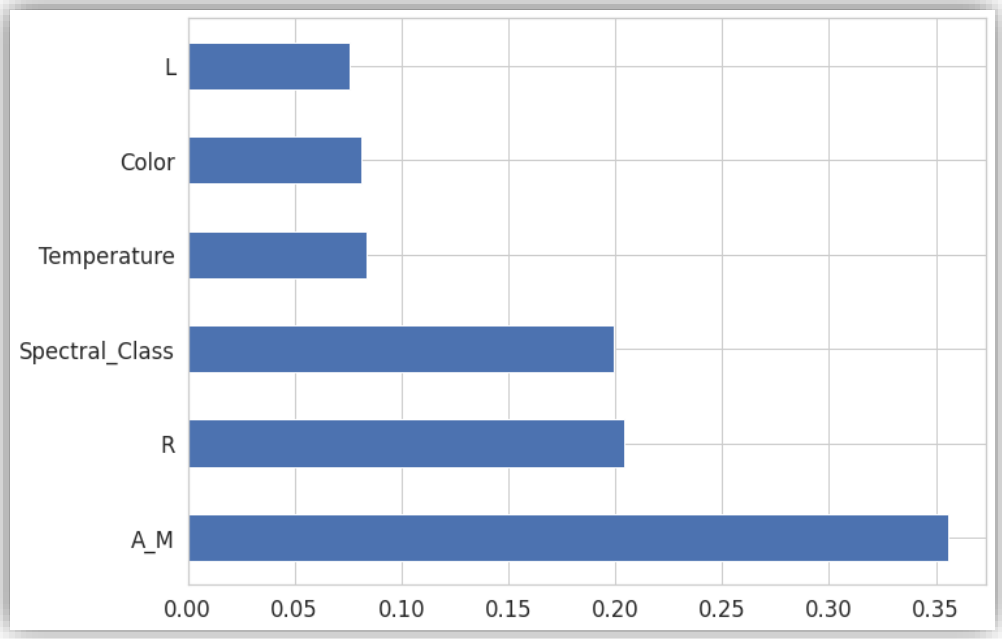
```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:3717: UserWarning: Th
  warnings.warn(msg)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:3723: UserWarning: Th
  warnings.warn(msg, UserWarning)
<seaborn.axisgrid.FacetGrid at 0x7f20a6b58890>
```



- Extratrees

  Melakukan ranking dengan bantuan package ExtraTreesClassifier.

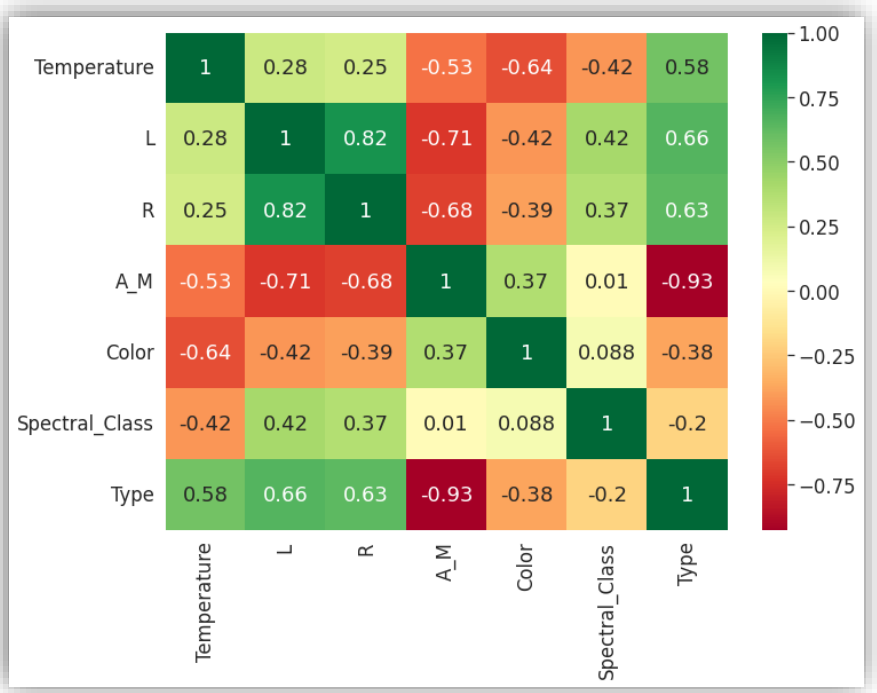
```
from sklearn.ensemble import ExtraTreesClassifier

model = ExtraTreesClassifier()
model.fit(x_fix,y_fix)

print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=x.columns)
feat_importances.nlargest(6).plot(kind='barh')
plt.show()
```




- Corrrelation Matrix Heatmap

  Melakukan ranking dengan bantuan korelasi matriks dan menampilkan hasilnya dengan heatmap.

```
x = data_out.iloc[:,0:8]  # independent columns
y = data_out.iloc[:,-1]    # target column

corrmat = data_out.corr()
top_corr_features = corrmat.index
# plt.figure(figsize=(50,50))
#plot heat map
g=sns.heatmap(data_out[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```




4. Feature Extraction
   - LDA

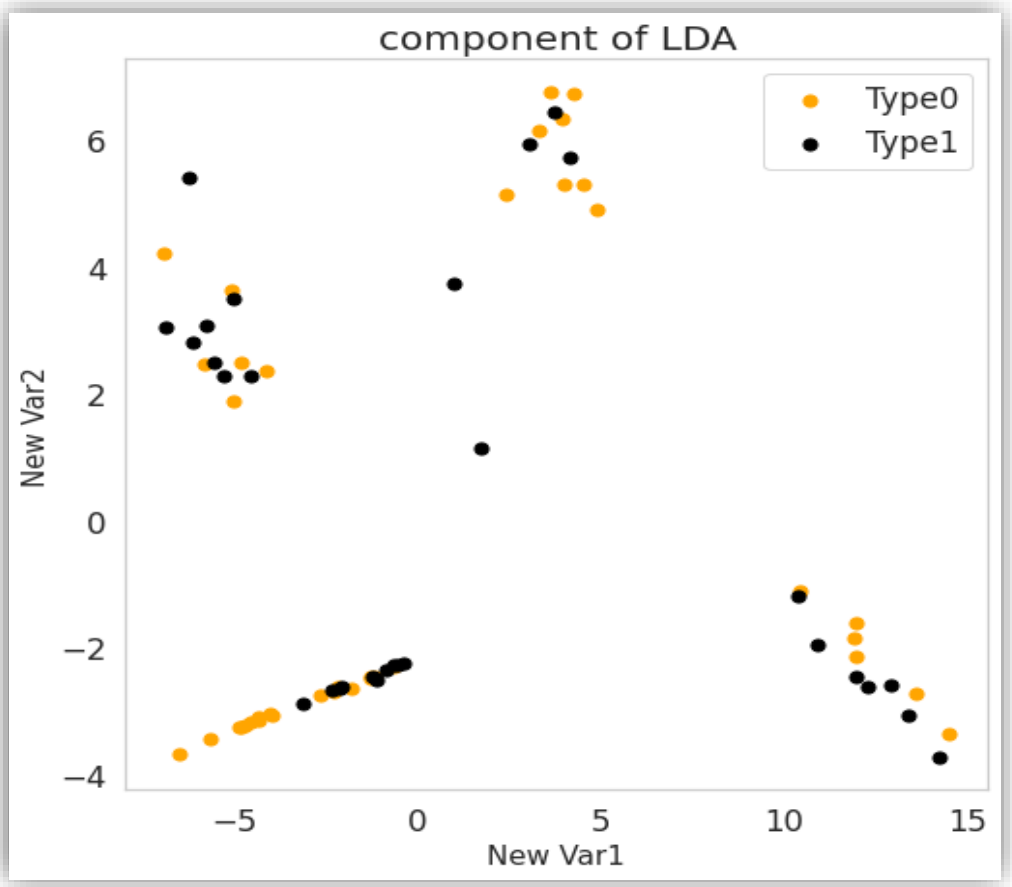
Membuat dua variabel baru yang akan digunakan untuk menampilkan hasil target yaitu kolom Type dengan menggunakan analisis diskriminan linear.

```
from sklearn import manifold, datasets, decomposition, discriminant_analysis

lda = discriminant_analysis.LinearDiscriminantAnalysis(n_components=2).fit_transform(x, y)

lda_data = pd.DataFrame(data = lda, columns = ['new_var1', 'new_var2'])
finalData = pd.concat([lda_data, y_fix], axis = 1)
finalData.head(11)
```

| | new_var1 | new_var2 | Type |
|---|---|---|---|
| 0 | -4.028459 | -3.046321 | 0.0 |
| 1 | -4.339476 | -3.112439 | 0.0 |
| 2 | -5.659834 | -3.438407 | 0.0 |
| 3 | -4.342571 | -3.146029 | 0.0 |
| 4 | -6.468529 | -3.691579 | 0.0 |
| 5 | -4.564296 | -3.184625 | 0.0 |
| 6 | -4.696125 | -3.239091 | 0.0 |
| 7 | -4.809984 | -3.266647 | 0.0 |
| 8 | -4.848460 | -3.267839 | 0.0 |
| 9 | -3.938749 | -3.077588 | 0.0 |
| 10 | -0.535529 | -2.273629 | 1.0 |

Menampilkan komponen dari proses LDA ini dengan menargetkan tipe 0 dan 1 oleh kedua variabel baru tersebut.

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('New Var1', fontsize = 15)
ax.set_ylabel('New Var2', fontsize = 15)
ax.set_title('component of LDA', fontsize = 20)
targets = [0.0, 1.0]
colors = ['orange', 'black']
for target, color in zip(targets,colors):
    indicesToKeep = finalData['Type'] == target
    ax.scatter(finalData.loc[indicesToKeep, 'new_var1']
               , finalData.loc[indicesToKeep, 'new_var2']
               , c = color
               , s = 50)
ax.legend(["Type0", "Type1"])
ax.grid()
```



- PCA

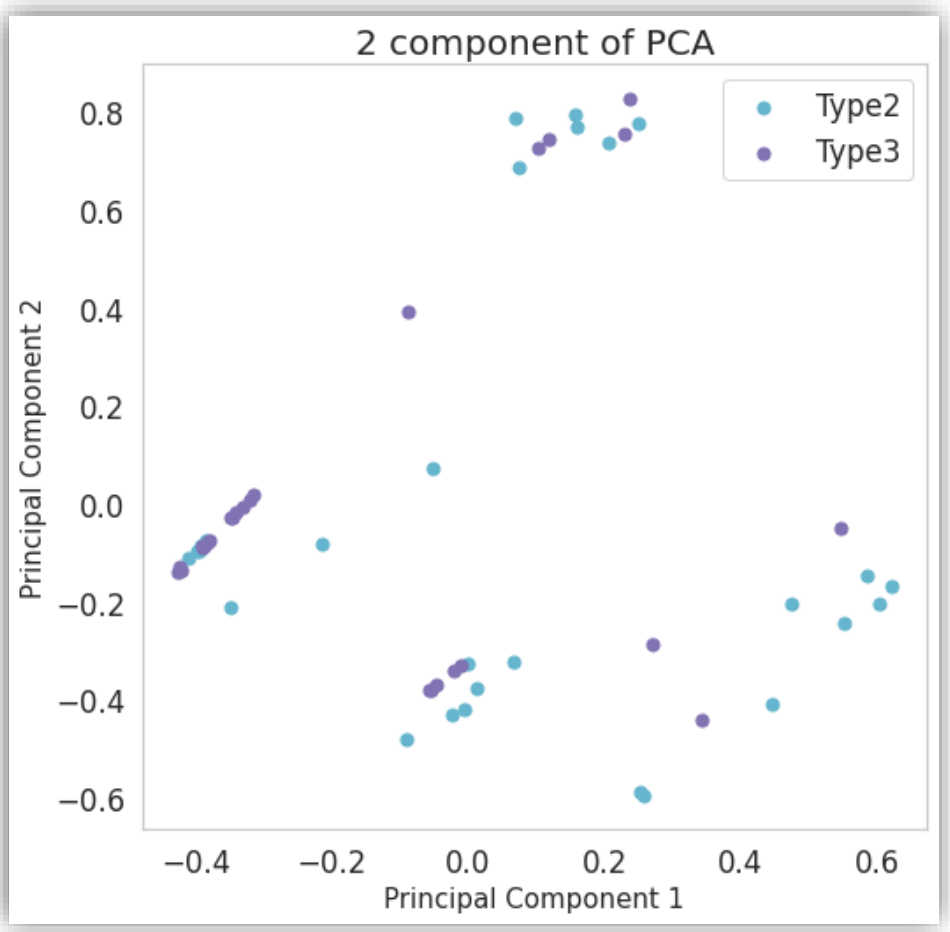
Membuat dua principal component baru yang akan digunakan untuk menampilkan hasil target yaitu kolom Type dengan menggunakan analisis PCA.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x_fix)
principalData = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2'])
finalData = pd.concat([principalData, y_fix], axis = 1)
finalData.head(11)
```

| | principal component 1 | principal component 2 | Type |
|---|---|---|---|
| 0 | -0.380385 | -0.072212 | 0.0 |
| 1 | -0.386400 | -0.080482 | 0.0 |
| 2 | -0.416408 | -0.118003 | 0.0 |
| 3 | -0.389720 | -0.082319 | 0.0 |
| 4 | -0.440060 | -0.143830 | 0.0 |
| 5 | -0.393203 | -0.087770 | 0.0 |
| 6 | -0.398349 | -0.092679 | 0.0 |
| 7 | -0.400912 | -0.095892 | 0.0 |
| 8 | -0.400945 | -0.096538 | 0.0 |
| 9 | -0.383693 | -0.072523 | 0.0 |
| 10 | -0.309669 | 0.022317 | 1.0 |

Menampilkan komponen dari proses PCA ini dengan menargetkan tipe 2 dan 3 oleh kedua komponen baru tersebut.

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component of PCA', fontsize = 20)
targets = [2.0, 3.0]
colors = ['c', 'm']
for target, color in zip(targets,colors):
    indicesToKeep = finalData['Type'] == target
    ax.scatter(finalData.loc[indicesToKeep, 'principal component 1']
               , finalData.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(["Type2", "Type3"])
ax.grid()
```




- t-SNE

Melakukan ekstraksi dengan membuat variabel x_2d yang akan melakukan fit terhadap dataset x_fix dengan bantuan package TSNE.

```
from sklearn.manifold import TSNE
tsne = TSNE(n_components=2, random_state=0)

x_2d = tsne.fit_transform(x_fix)
```

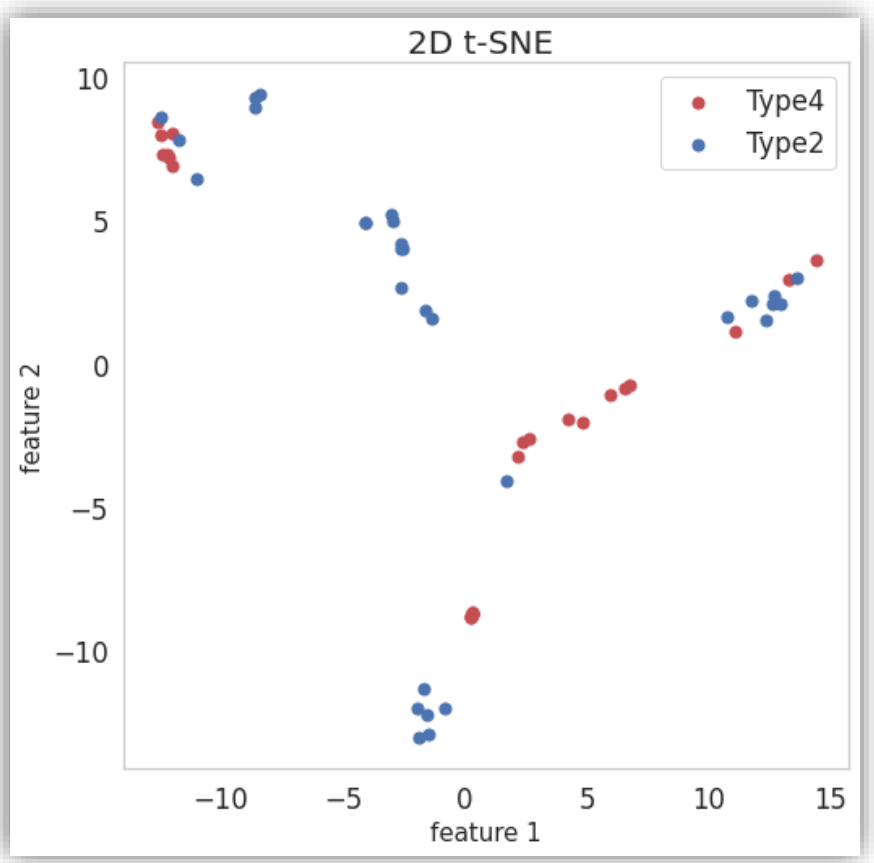
Membuat dua fitur baru yang akan digunakan untuk menampilkan hasil target yaitu kolom Type.

```
new_data = pd.DataFrame(data = x_2d, columns = ['feature 1', 'feature 2'])
tsne_data = pd.concat([new_data, y_fix], axis = 1)
tsne_data.head(11)
```

| | feature 1 | feature 2 | Type |
|---|---|---|---|
| 0 | 10.775083 | 1.262862 | 0.0 |
| 1 | 11.420237 | 1.563958 | 0.0 |
| 2 | 13.852415 | 3.267747 | 0.0 |
| 3 | 11.229586 | 2.079783 | 0.0 |
| 4 | 14.482038 | 3.890038 | 0.0 |
| 5 | 11.765959 | 2.334239 | 0.0 |
| 6 | 12.053390 | 2.673248 | 0.0 |
| 7 | 12.359558 | 2.746352 | 0.0 |
| 8 | 12.489676 | 2.727125 | 0.0 |
| 9 | 10.524278 | 1.498966 | 0.0 |
| 10 | 2.249708 | -2.957296 | 1.0 |

Menampilkan fitur dari proses t-SNE ini dengan menargetkan tipe 4 dan 2 oleh kedua fitur baru tersebut.

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('feature 1', fontsize = 15)
ax.set_ylabel('feature 2', fontsize = 15)
ax.set_title('2D t-SNE ', fontsize = 20)
targets = [4.0, 2.0]
colors = ['r', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = tsne_data['Type'] == target
    ax.scatter(tsne_data.loc[indicesToKeep, 'feature 1']
               , tsne_data.loc[indicesToKeep, 'feature 2']
               , c = color
               , s = 50)
ax.legend(["Type4", "Type2"])
ax.grid()
```




Source Code:

https://github.com/trueazp/Data-Mining/blob/main/assignments/assignment04/Tugas04_Pertemuan5_PDM_H071191035.ipynb