Final Projek Akhir

FOR

Data Mining

KELOMPOK 4 C1

ANGGOTA KELOMPOK

M NOVIL FAHLEVY
2109116095

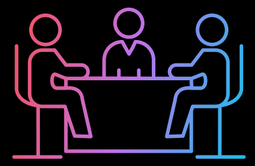
UTARI W ARDHANA
2109116103

ARDITA DYAH P
2109116089

LINK COLLAB & DATA STUDIO

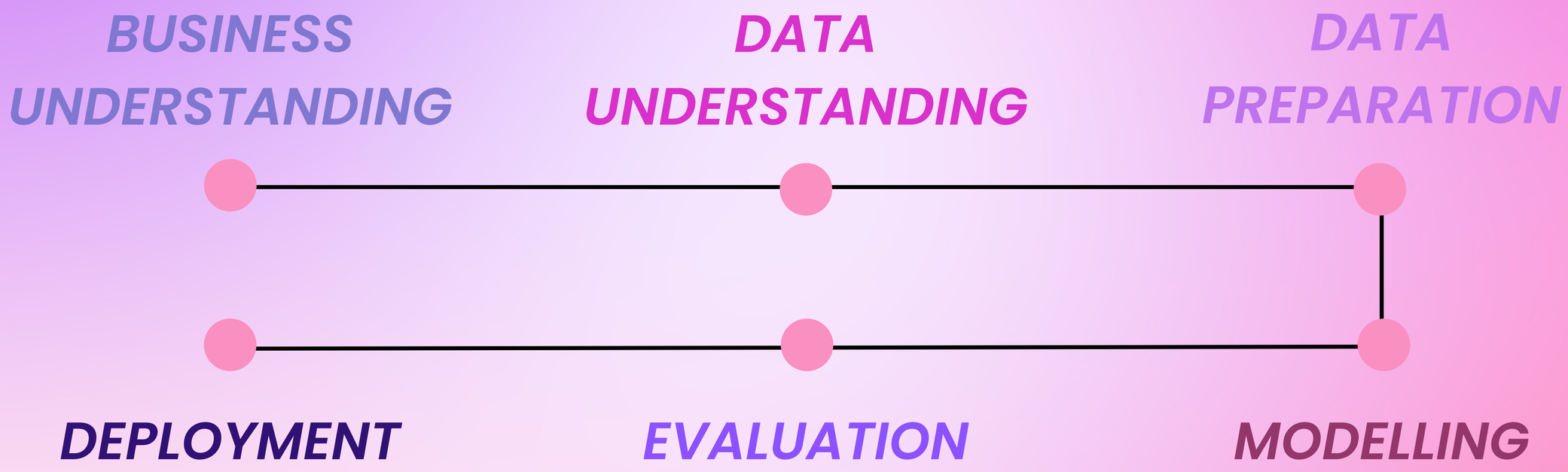
Collab :
https://colab.research.google.com/drive/1elBpFKhvDBRNdc4AF1z3LaOkQIoOfU1V?usp=sharing

Data Studio :
https://lookerstudio.google.com/reporting/8202e098-bed9-4ce2-8736-f0738f92c8d2/page/p_2um3yenv5c



Supervised

ALUR CRISP-DM



TAHAP 1
BUSINESS UNDERSTANDING

BUSINESS BACKGROUND

Sebuah perusahaan Real Estate tentunya membutuhkan data-data perumahan yang akan dipantau dan diperjual-belikan kepada orang yang memerlukan, tidak hanya itu data data juga harus disimpan dan diolah di tempat yang benar agar tidak terjadi redudansi ataupun duplikat Oleh karena itu, perusahaan Real Estate harus memiliki bukti nyata kepemilikan perumahan mereka.

VISI

Mengembangkan sebuah grup bisnis properti dengan semangat yang unggul dan penuh inovasi, sehingga menciptakan nilai tambah dalam menyediakan kehidupan yang lebih baik bagi masyarakat dan memberikan kemakmuran dan kesejahteraan bagi para pemangku kepentingan.

MISI

Menjadi yang terdepan dalam bisnis properti dengan menjadi yang paling unggul, profesional dan menguntungkan, sehingga menjadi pilihan pertama bagi para konsumen, menjadi tempat kerja yang paling menarik dan menantang bagi para karyawan, menjadi investasi yang paling menguntungkan bagi para pemegang saham dan menjadi berkat yang nyata bagi masyarakat dan Tanah Air

BUSINESS BACKGROUND

TARGET

Milennial atau orang yang membutuhkan rumah atau properti dengan harga murah dan ukuran yang minimalis untuk keluarga kecil namun fasilitas yang lengkap

ALASAN MENGGUNAKAN DATA MINING

Dengan menggunakan data mining, perusahaan dapat lebih mudah untuk Mengelompokkan dan mengklasifikasikan berbagai jenis rumah, ukuran, properti dan ciri atau target customer yang sesuai dengan marketing real estate

ACCESS SITUATION

1. Tools yang digunakan :

* Google Colab
* Google Drive
* Kaggle
* Google Data Studio

BUSINESS BACKGROUND

ACCESS SITUATION

2. Cost Vs Benefit Anlyze

* Keuntungan
    * Dapat mengelompokkan dan mengklasifikasikan tipe dan harga rumah dengan mudah sehingga gampang untuk diperjual-belikan
    * Memudahkan customer untuk memilih tipe rumah dan menentukan sesuai budget yang customer punya
    * Meminimalisir kerugian yang dapat dihasilkan seperti tidak kesesuaian bduget dengan tipe rumah
    * Perusahaan dapat mengelompokkan customer sesuai dengan tipe rumah

* Kelebihan
    * Perusahaan hanya perlu untuk memperbaharui melalui data set tanpa harus merubah dari tools nya
    * Data yang disimpan akan tetap rapi dan bisa dicari dengan mudah
    * Sudah dilengkapi library yang akan memudahkan perusahaan untuk mengklasifikasi, regresi, klastering dan lain lain agar data yang dihasilkan itu sesuai.

TAHAP 2
DATA UNDERSTANDING

DESCRIBE DATA

ATTRIBUTE INFORMATION

1. Price = harga rumah
2. Area = dimana area rumah
3. Bedrooms = jumlah bedrooms pada tiap rumah
4. Bathrooms = jumlah bathrooms pada tiap rumah
5. Stories = jumlah stories pada tiap rumah
6. Mainroad = rumah yang berada di main road dengan penjelasan "yes" or "no"
7. Guestroom = rumah yang memiliki guestroom dengan penjelasan "yes" or "no"
8. Basement = rumah yang memiliki basement dengan penjelasan "yes" or "no"
9. Hotwaterheating = rumah yang memiliki hotwaterheating dengan penjelasan "yes" or "no"
10. Aircondtioning = rumah yang memiliki airconditioning dengan penjelasan "yes" or "no"
11. Parking = jumlah area parking tiap rumah
12. Prefarea
13. Furnishingstatus = penjelasan tiap rumah apakah "furnished", "semi-furnished", dan "unfurnished"

DESCRIBE DATA

```
df.info()
```

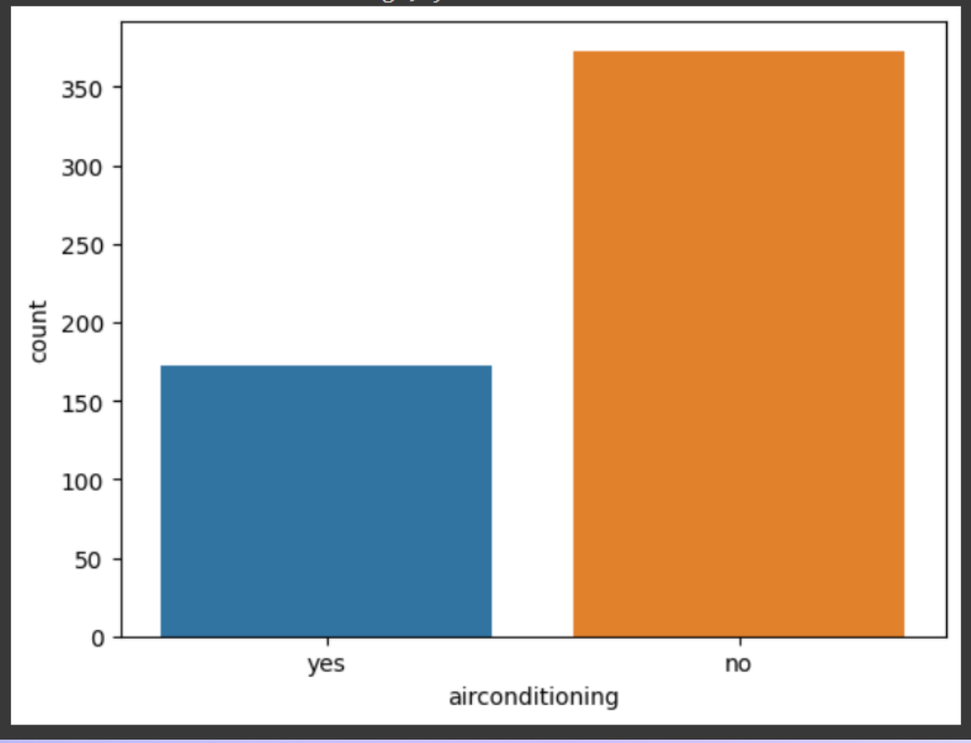
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    int64
 2   bedrooms          545 non-null    int64
 3   bathrooms         545 non-null    int64
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

Dataset ini memiliki target harga (price), beberapa data yang memiliki atribute boolean, dan data yang bersifat kategorikal.

EXPLORE DATA

```
1 #@title Perbandingan Rumah Menggunakan airconditioning dan Tidak
2 sns.countplot(data=df,x="airconditioning")
```

```
<Axes: xlabel='airconditioning', ylabel='count'>
```

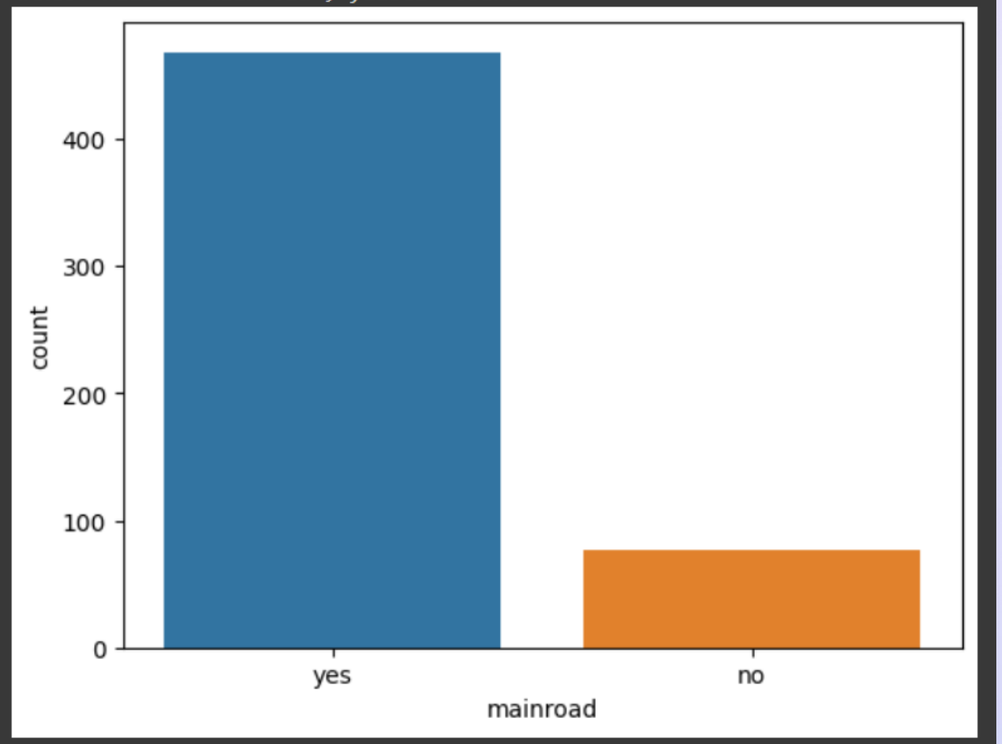


Disini terlihat lebih banyak rumah yang tidak memiliki pendingin ruangan.

EXPLORE DATA

```
[ ]  1 #@title Perbandingan Rumah yang berada di mainroad dan tidak
     2 sns.countplot(data=df,x="mainroad")
```

```
<Axes: xlabel='mainroad', ylabel='count'>
```



Lebih banyak rumah yang terhubung ke jalan utama.

TAHAP 3
DATA PREPARATION

CLEAN DATA

```
# Merubah Datatype mainroad, guestroom, basement, hotwaterheating, airconditioning,
# dan prefarea jadi 1,0 karena isi datanya hanya yes dan no

df2.mainroad = df.mainroad.map({"yes":1,"no":0})
df2.guestroom = df.guestroom.map({"yes":1,"no":0})
df2.basement = df.basement.map({"yes":1,"no":0})
df2.hotwaterheating = df.hotwaterheating.map({"yes":1,"no":0})
df2.airconditioning = df.airconditioning.map({"yes":1,"no":0})
df2.prefarea = df.prefarea.map({"yes":1,"no":0})
```

CLEAN DATA

```python
# Kategorikan price menjadi 3 tipe
def categorize_price(price):
    if price <= 5000000:
        return 'cheap'
    elif price > 5000000 and price <= 8000000:
        return 'medium'
    else:
        return 'expensive'

df2.insert(loc=1, column='pricecategory', value=df2['price'].apply(categorize_price))

df2.head()
```

	price	pricecategory	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	aircond
0	13300000	expensive	7420	4	2	3	1	0	0	0	
1	12250000	expensive	8960	4	4	4	1	0	0	0	
2	12250000	expensive	9960	3	2	2	1	0	1	0	
3	12215000	expensive	7500	4	2	2	1	0	1	0	
4	11410000	expensive	7420	4	1	2	1	1	1	0	

MISSING VALUE

```
df2.isna().sum()
```

```
price             0
pricecategory     0
area              0
bedrooms          0
bathrooms         0
stories           0
mainroad          0
guestroom         0
basement          0
hotwaterheating   0
airconditioning   0
parking           0
prefarea          0
furnishingstatus  0
dtype: int64
```

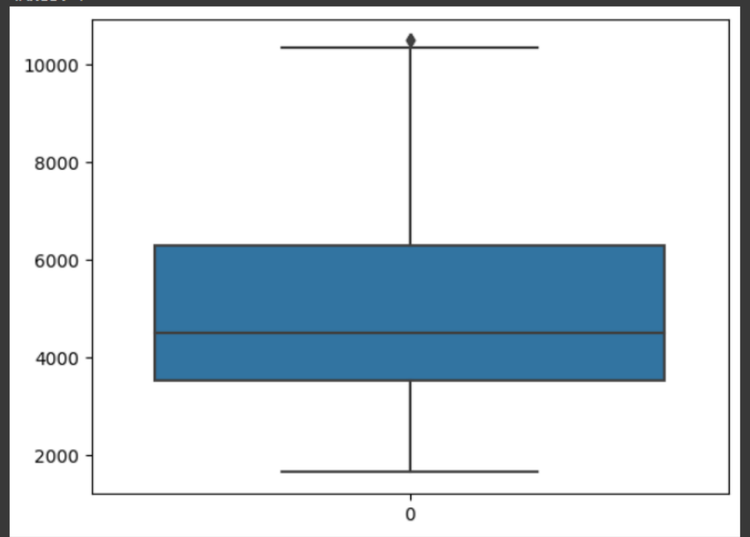
Tidak ada missing value di dataset ini.

OUTLIER

```
1 # Karena ga diatas 5 persen kita bisa hapus data nya
2 df2 = df2[~((df2.area > ba) | (df2.area < bw))]
```

```
[ ]  1 sns.boxplot(df2.area)
```

<Axes: >

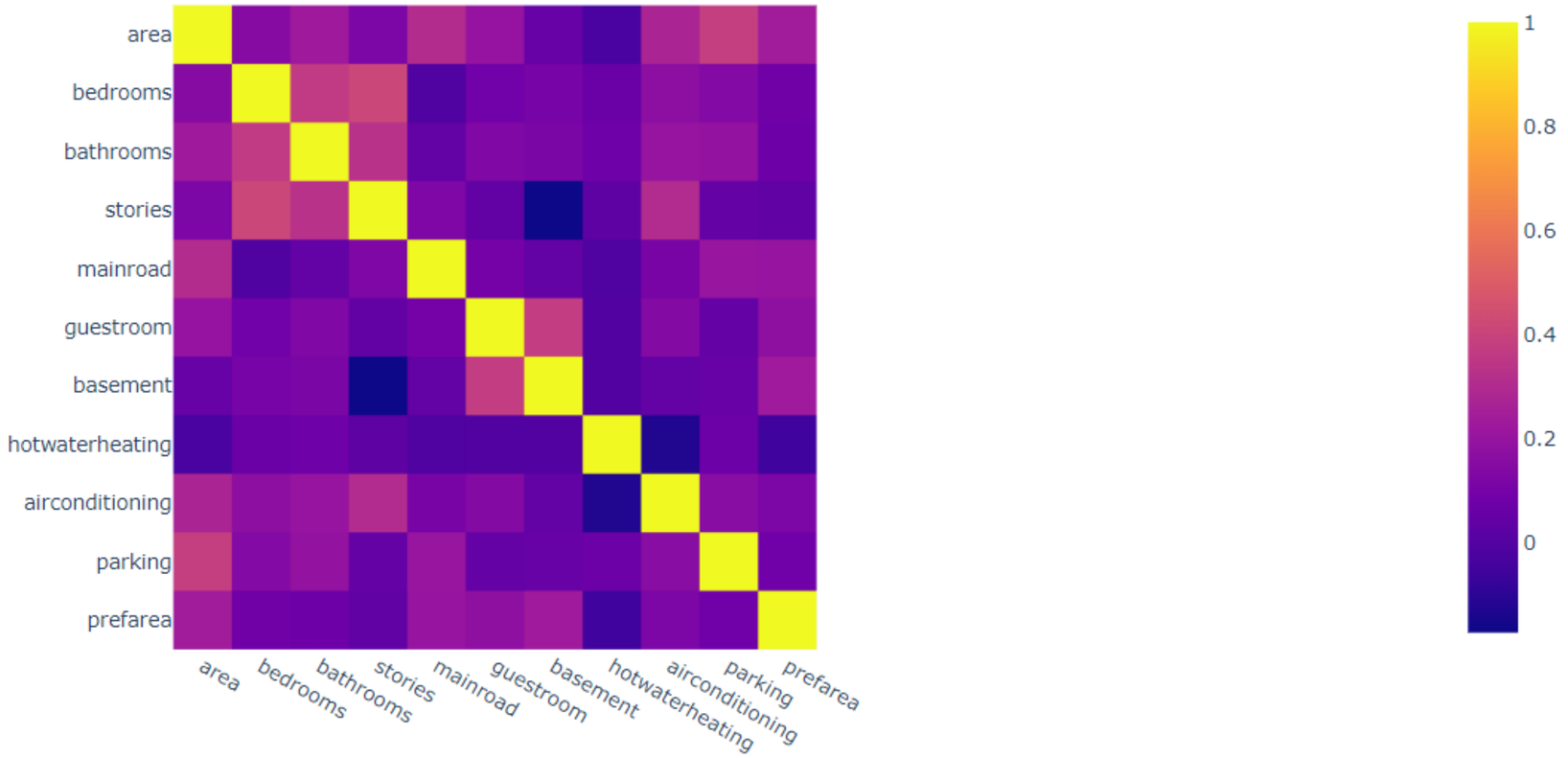



```
[95] len(outlier) / (len(df2) * 100)

     0.00022018348623853212
```

Ini merupakan outlier dari kolom area.

VISUALISASI KORELASI

```
1 px.imshow(df3.corr())
```



CONSTRUCT DATA

```
df3 = df2.drop(columns=["price"])
df3.head()
```

	pricecategory	area	bedrooms	bathrooms	stories
0	expensive	7420	4	2	3
1	expensive	8960	4	4	4
2	expensive	9960	3	2	2
3	expensive	7500	4	2	2
4	expensive	7420	4	1	2

Disini kolom price tidak diperlukan karena sudah ada kolom pricecategory.

Encoding

```python
# Kategorikan price menjadi 3 tipe
def convert_pricecategory_to_numeric(pricecategory):
    if pricecategory == 'cheap':
        return 1
    elif pricecategory == 'medium':
        return 2
    else:
        return 3

df3['pricecategory'] = df3['pricecategory'].apply(convert_pricecategory_to_numeric)

df3.head()
```

	pricecategory	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterhe
0	3	7420	4	2	3	1	0	0	
1	3	8960	4	4	4	1	0	0	
2	3	9960	3	2	2	1	0	1	
3	3	7500	4	2	2	1	0	1	
4	3	7420	4	1	2	1	1	1	

Kolom pricecategory diencoding menjadi numerical

Encoding

```
# Kategorikan price menjadi 3 tipe
def convert_pricecategory_to_numeric(pricecategory):
    if pricecategory == 'cheap':
        return 1
    elif pricecategory == 'medium':
        return 2
    else:
        return 3

df3['pricecategory'] = df3['pricecategory'].apply(convert_pricecategory_to_numeric)

df3.head()
```

	pricecategory	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterhe
0	3	7420	4	2	3	1	0	0	
1	3	8960	4	4	4	1	0	0	
2	3	9960	3	2	2	1	0	1	
3	3	7500	4	2	2	1	0	1	
4	3	7420	4	1	2	1	1	1	

Kolom furnishingstatus diencoding sehingga menjadi 3 bagian.

FORMAT DATA/ PREPROCESSING

```python
# Kategorikan price menjadi 3 tipe
def convert_pricecategory_to_numeric(pricecategory):
    if pricecategory == 'cheap':
        return 1
    elif pricecategory == 'medium':
        return 2
    else:
        return 3

df3['pricecategory'] = df3['pricecategory'].apply(convert_pricecategory_to_numeric)

df3.head()
```

FORMAT DATA/ PREPROCESSING

```
df4 = pd.get_dummies(df3, columns=['furnishingstatus'])
df4
```

```
# membuat variabel untuk train split test
TARGET = df4['pricecategory']
data = df4.drop(columns='pricecategory')
```

Split train test

```
# membuat variabel untuk train split test
TARGET = df4['pricecategory']
data = df4.drop(columns='pricecategory')
```

Selanjutnya adalah membuat visualisasi Train Test Split dengan tujuannya membagi data untuk e

Membagi data menjadi data train dan data test, disini targetnya adalah pricecategory

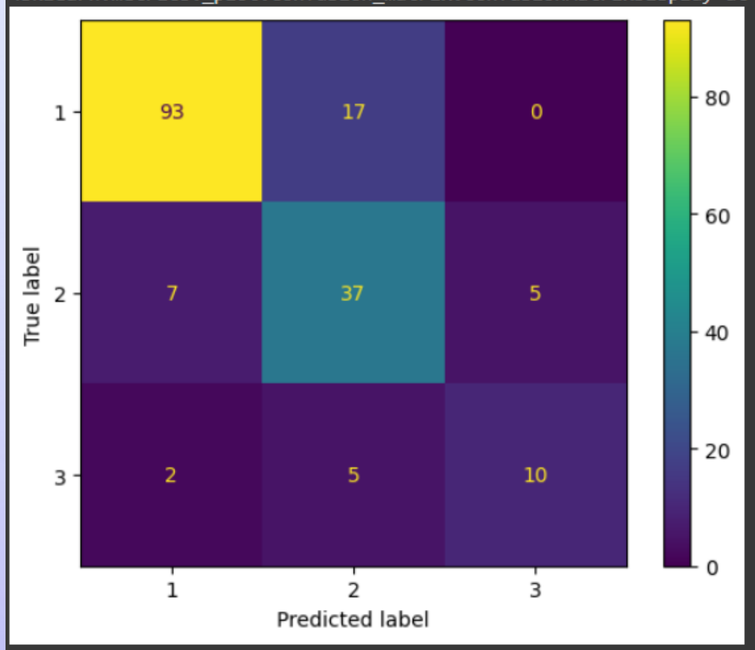
TAHAP 4&5
MODELLING & EVALUATION

NAIVE BAYES

```
1 cm = confusion_matrix(y_test, nb_pred, labels=modelnb.classes_)
```

```
1 disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=modelnb.classes_)
2 disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f29e8399750>
```




```
1 TN = cm[0][0]
2 FP = cm[0][1]
3 FN = cm[1][0]
4 TP = cm[1][1]
```

NAIVE BAYES

Precision

```
[210] precision = TP / (TP + FP)
      print(precision * 100)
```

```
75.55555555555556
```

Recall

```
[211] recall = TP / (TP + FN)
      print(recall * 100)
```

```
70.83333333333334
```

F1

```
[212] f1 = (2 * precision * recall) / (precision + recall)
      print(f1 * 100)
```

```
73.11827956989248
```

Accuracy

```
[213] accuracy = (TP + TN) / (TP + FN + TN + FP)
      print(accuracy * 100)
```

```
84.07643312101911
```

Specificity

```
specificity = TN / (TN + FP)
print(specificity * 100)
```

```
89.90825688073394
```

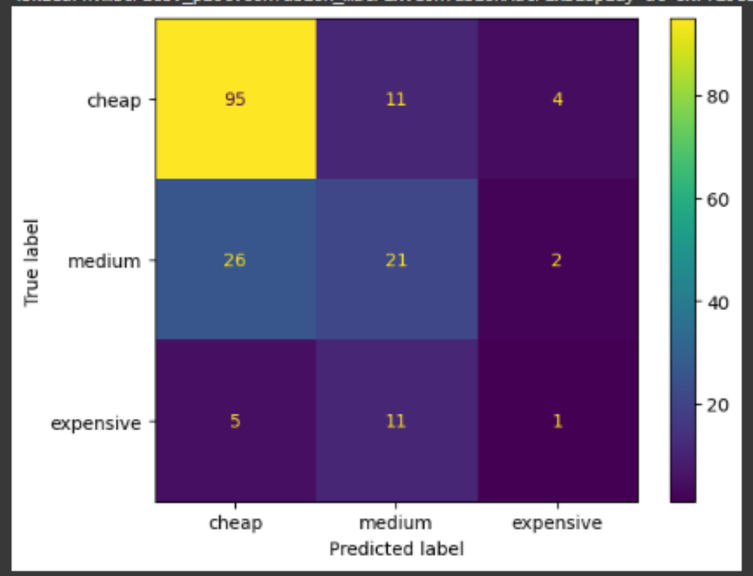
KNN

Confusion Matrix

```
[ ]  1 cm = confusion_matrix(y_test, predictions)
```

```
[ ]  1 disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['cheap', 'medium', 'expensive'])
     2 disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f29e82c5510>
```




```
[ ]  1 TN = cm[0][0]
     2 FP = cm[0][1]
     3 FN = cm[1][0]
     4 TP = cm[1][1]
```

KNN

Precision

```
[126] precision = TP / (TP + FP)
      print(precision * 100)
```

```
65.625
```

Recall

```
[127] recall = TP / (TP + FN)
      print(recall * 100)
```

```
44.680851063829785
```

F1

```
[128] f1 = (2 * precision * recall) / (precision + recall)
      print(f1 * 100)
```

```
53.16455696202532
```

Accuracy

```
[129] accuracy = (TP + TN) / (TP + FN + TN + FP)
      print(accuracy * 100)
```

```
75.81699346405229
```

Specificity

```
[130] specificity = TN / (TN + FP)
      print(specificity * 100)
```

```
89.62264150943396
```

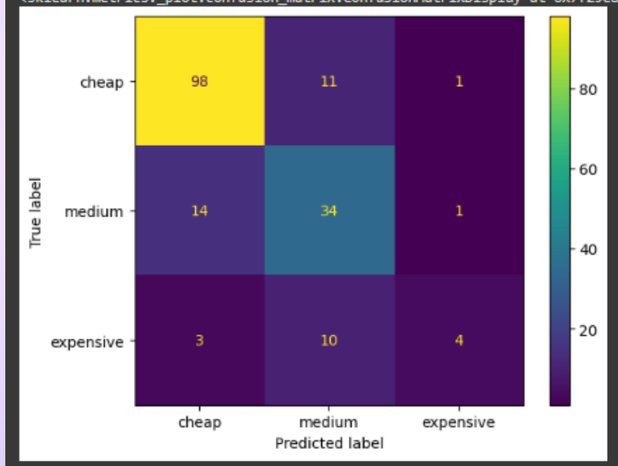
C4.5

```python
def addition(n):
    return n + 1

dtc_pred = list(map(addition, dtc_pred))
```

```python
cm = confusion_matrix(y_test, dtc_pred)
```

```python
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['cheap', 'medium', 'expensive'])
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f29e8187f10>
```




```python
TN = cm[0][0]
FP = cm[0][1]
FN = cm[1][0]
TP = cm[1][1]
```

C4.5

Precision

```
precision = TP / (TP + FP)
print(precision * 100)
```

```
75.55555555555556
```

Recall

```
[139] recall = TP / (TP + FN)
      print(recall * 100)
```

```
70.83333333333334
```

F1

```
[140] f1 = (2 * precision * recall) / (precision + recall)
      print(f1 * 100)
```

```
73.11827956989248
```

Accuracy

```
accuracy = (TP + TN) / (TP + FN + TN + FP)
print(accuracy * 100)
```

```
84.07643312101911
```

Specificity

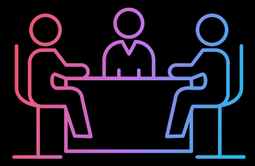
```
[142] specificity = TN / (TN + FP)
      print(specificity * 100)
```

```
89.90825688073394
```

TAHAP 6
DEPLOYMENT

DEPLOYMENT

```
# Export df yang sebelum encoding
# df3 dipakai karena ini df yang setelah cleaning dan sebelum encoding
df3.to_csv("output.csv", index=False)
```



Unsupervised

LIBRARY

```
[ ]
# Untuk data dan analisis
import pandas as pd

# Melakukan pehritungan
import numpy as np
import math as m

# Membuat sebuah visualisasi
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
```

```
[ ]
#Link awal gdrive
link = "https://drive.google.com/file/d/1D9lHApdyL2Kn2u3dUOhH4loy4EW8T0N-/view?usp=share_link"

#Setelah Diubah
URL = "https://drive.google.com/uc?id=1D9lHApdyL2Kn2u3dUOhH4loy4EW8T0N-"
```

DATA UNDERSTANDING

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 17 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   BALANCE                           8950 non-null   float64
 1   BALANCE_FREQUENCY                 8950 non-null   float64
 2   PURCHASES                         8950 non-null   float64
 3   ONEOFF_PURCHASES                  8950 non-null   float64
 4   INSTALLMENTS_PURCHASES            8950 non-null   float64
 5   CASH_ADVANCE                      8950 non-null   float64
 6   PURCHASES_FREQUENCY               8950 non-null   float64
 7   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 8   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
 9   CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 10  CASH_ADVANCE_TRX                  8950 non-null   int64
 11  PURCHASES_TRX                     8950 non-null   int64
 12  CREDIT_LIMIT                      8949 non-null   float64
 13  PAYMENTS                          8950 non-null   float64
 14  MINIMUM_PAYMENTS                  8637 non-null   float64
 15  PRC_FULL_PAYMENT                  8950 non-null   float64
 16  TENURE                            8950 non-null   int64
dtypes: float64(14), int64(3)
memory usage: 1.2 MB
```

DESCRIBE

- Cust_ID : id kostumer
- Balance : Jumlah saldo pada kartu kredit
- Balance_frequency : Frekuensi saldo
- Purchases : Pembelian
- Oneoff_Purchases : Salah satu pembelian
- Installments_Purchases :Pembelian
- Cash_Advance : Pemasukan
- Purchases_Frequency : Frekuensi Pembelian
- Oneoff_Purchases_Frequency : Frekuensi Salah satu pembelian
- Purchases_Installments_Frequency : Frekuensi install pembela
- Cash_Advance_Frequency : Frekuensi Pemasukan
- Cash_Advance_TRX:
- Purchases_TRX:
- Credit_limit: Limir kartu kredit
- Payments: Pembayaran
- Minimum_payments: Pembayaran minimum
- PRC_full_payments:
- Tenure: Masa jabatan

DATA CLEANING

Drop unused column

```
df = df.drop('CUST_ID', axis=1)
df
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES
0	40.900749	0.818182	95.40
1	3202.467416	0.909091	0.00
2	2495.148862	1.000000	773.17
3	1666.670542	0.636364	1499.00
4	817.714335	1.000000	16.00
...	...	...	...
8945	28.493517	1.000000	291.12
8946	19.183215	1.000000	300.00
8947	23.398673	0.833333	144.40
8948	13.457564	0.833333	0.00
8949	372.708075	0.666667	1093.25

Handling Missing Values

```
[152] df.isnull().any()

BALANCE                             False
BALANCE_FREQUENCY                   False
PURCHASES                           False
ONEOFF_PURCHASES                    False
INSTALLMENTS_PURCHASES              False
CASH_ADVANCE                        False
PURCHASES_FREQUENCY                 False
ONEOFF_PURCHASES_FREQUENCY          False
PURCHASES_INSTALLMENTS_FREQUENCY    False
CASH_ADVANCE_FREQUENCY              False
CASH_ADVANCE_TRX                    False
PURCHASES_TRX                       False
CREDIT_LIMIT                         True
PAYMENTS                            False
MINIMUM_PAYMENTS                     True
PRC_FULL_PAYMENT                    False
TENURE                              False
dtype: bool
```

```
df['MINIMUM_PAYMENTS'] = df['MINIMUM_PAYMENTS'].fillna(df['MINIMUM_PAYMENTS'].mean())
df['CREDIT_LIMIT'] = df['CREDIT_LIMIT'].fillna(df['CREDIT_LIMIT'].mean())
```

METRIC PENGUKURAN

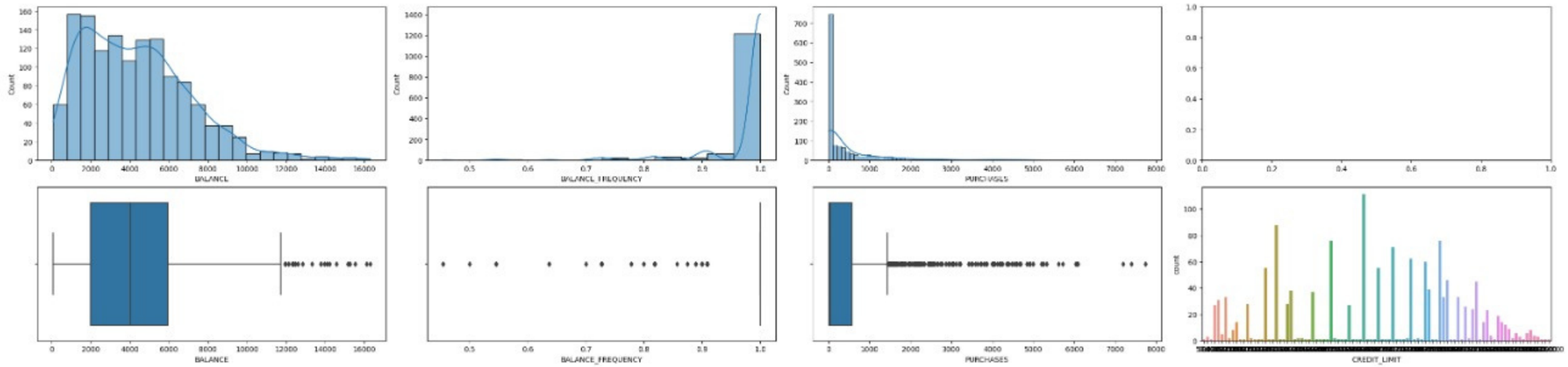
```python
# https://stackoverflow.com/questions/28075699/coloring-cells-in-pandas
best = [
    df_metric.iloc[:,0].max() # Makin Tinggi Nilai silhoutte makin bagus
    ,df_metric.iloc[:,1].max(), # Makin Tinggi Nilai CH makin bagus
    df_metric.iloc[:,2].min()
    ] # Makin rendah nilai David Bouldin makin bagus

#Ini untuk conditional formatting
def _color_green(val):
    if val in best:
        return 'background-color: green; color:white'
    return 'backround-color:white'
```

```python
# Menampilkan tabel setelah dilakukan contional formatting
df_metric.style.applymap(_color_green)
```

	silhouette_score	calinski_harabasz_score	davies_bouldin_score
AgglomerativeClustering	0.158069	1.588442	1161.990892
KMeans	0.193169	1.549390	1482.159003
Birch	0.149559	1.596233	1159.906763

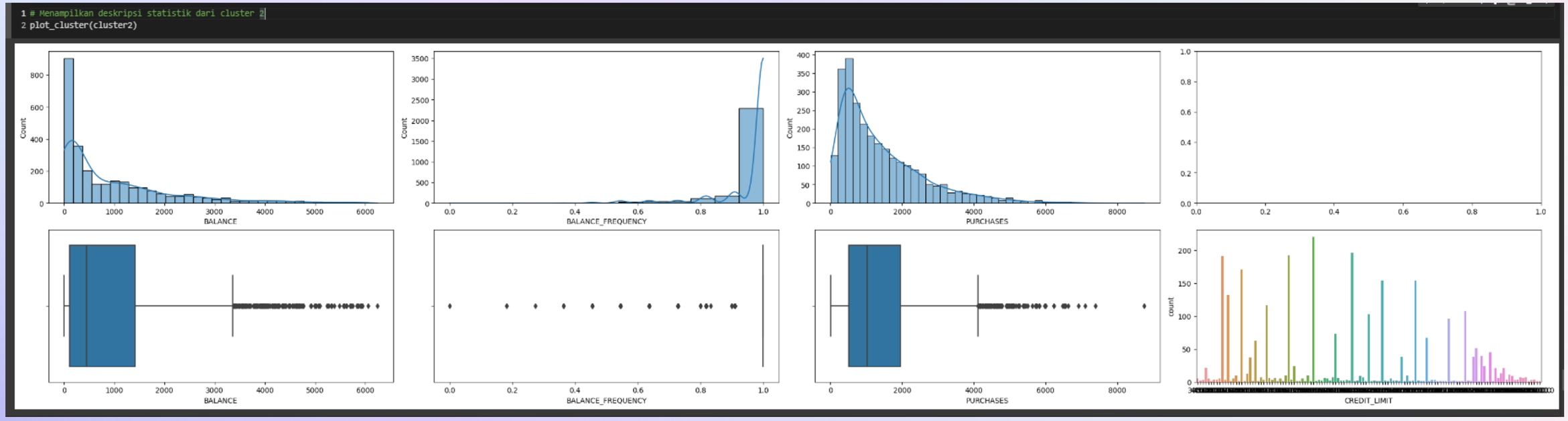
CLUSTER 1

```
[ ]  1  # Menampilkan distribusi dari cluster1
     2  plot_cluster(cluster1)
```



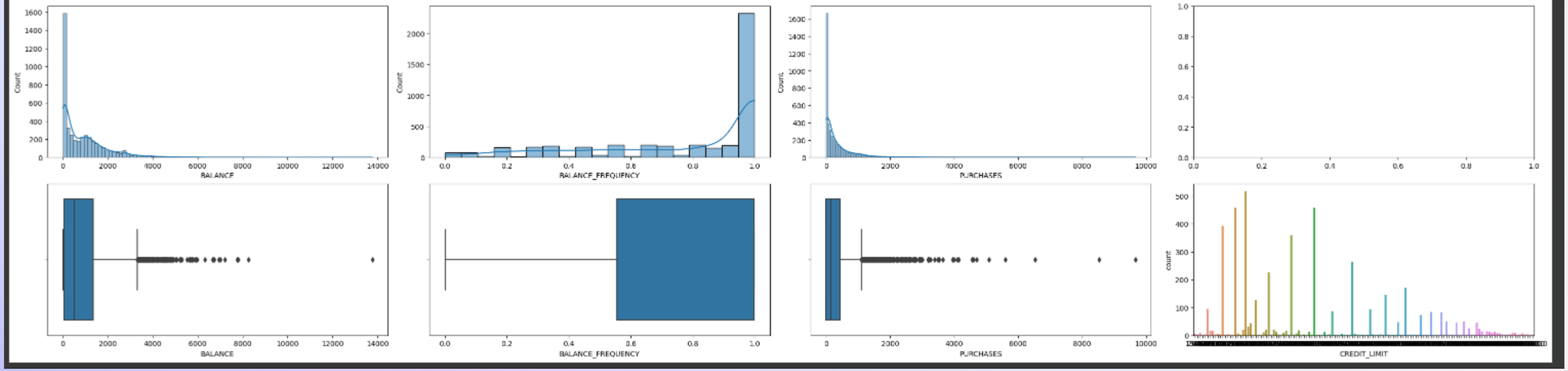
CLUSTER 2

```
# Menampilkan deskripsi statistik dari cluster 2
plot_cluster(cluster2)
```



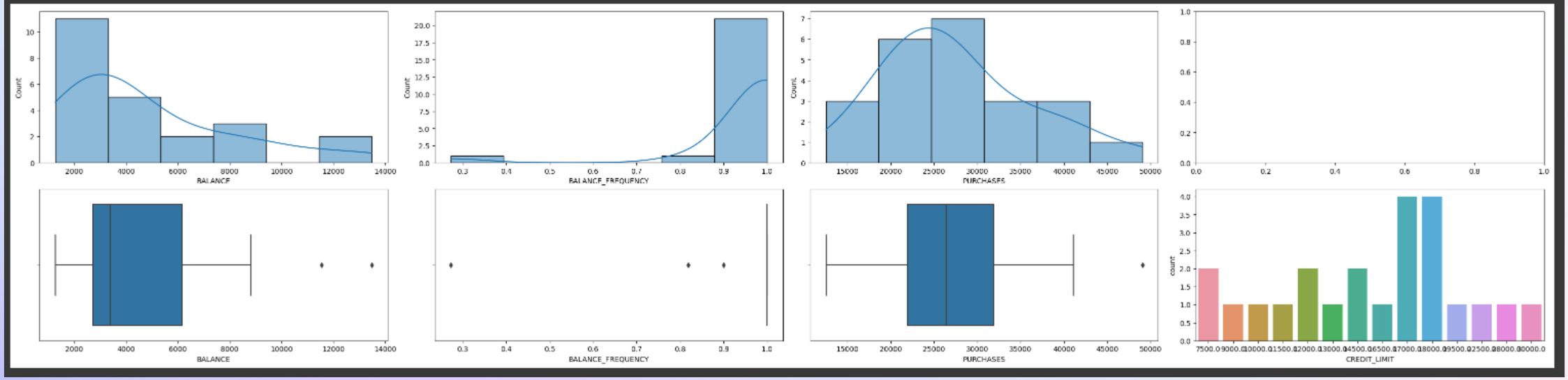
CLUSTER 3

```
# Menampilkan distribusi dari cluster3
plot_cluster(cluster3)
```



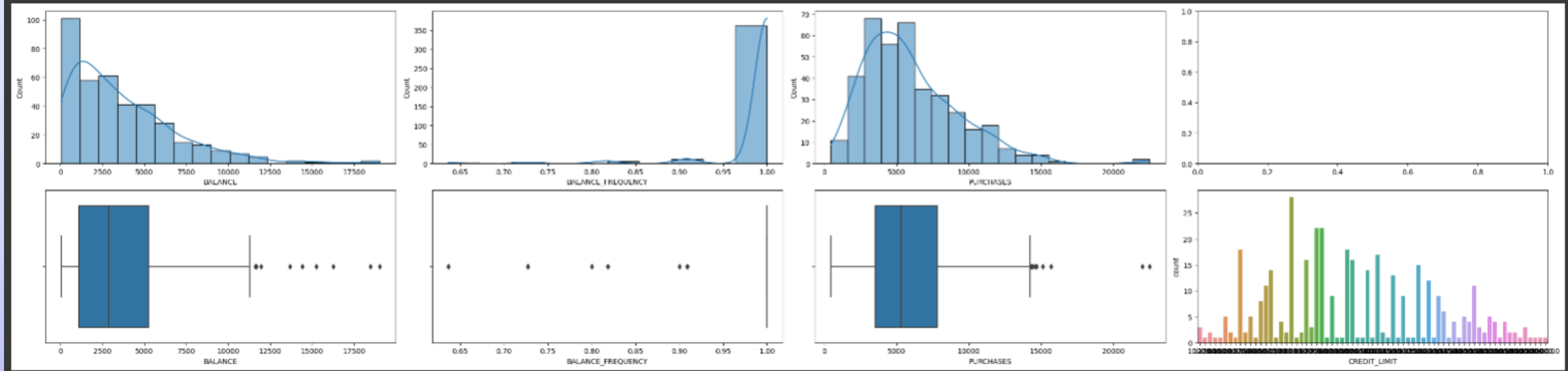
CLUSTER 4

```
# Menampilkan distribusi dari cluster 4
plot_cluster(cluster4)
```



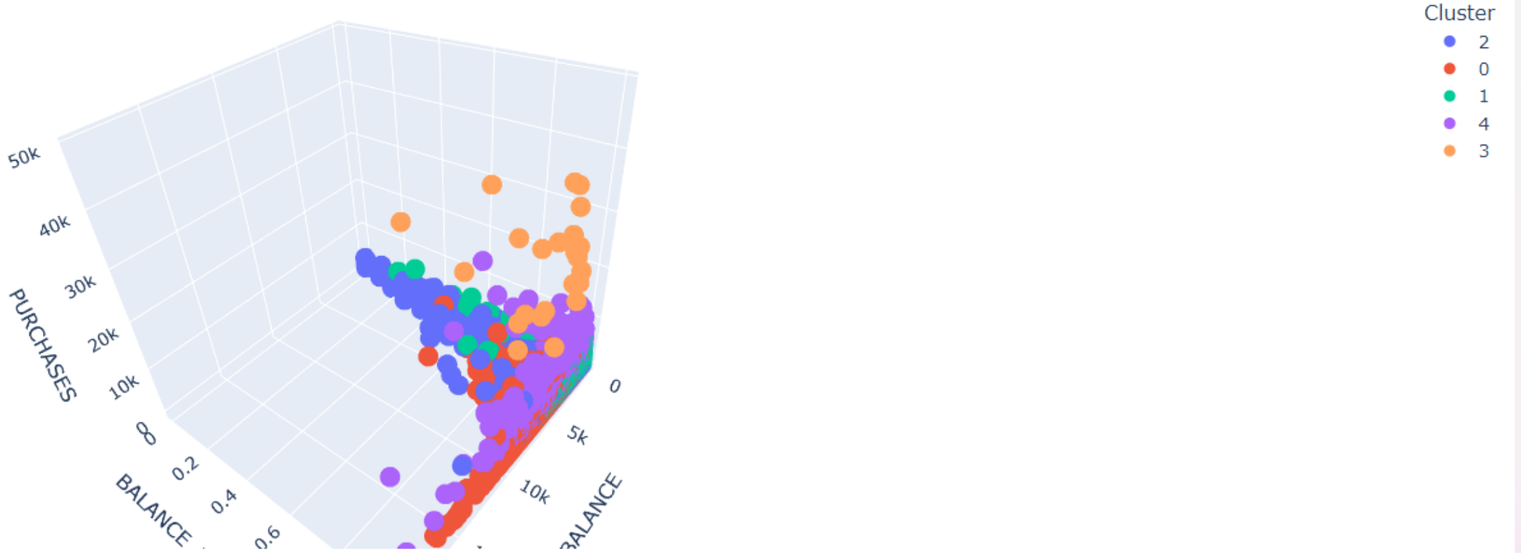
CLUSTER 5

```
# Menampilkan distribusi dari cluster 4
plot_cluster(cluster5)
```



KESIMPULAN

```
1 # TODO : DESKRIPSI DARI TIAP CLUSTER
2 br.plot_3d(cols[0],cols[1],cols[2])
```



DEPLOYMENT

```
# Export df yang sebelum encoding
# df3 dipakai karena ini df yang setelah cleaning dan sebelum encoding
df.to_csv("output.csv", index=False)
```

Thank you.

+123-456-7890

hello@reallygreatsite.com