

# Final Project

Kelompok 6

DATA MINING

START

# Our Team



WINDY WIDYSTUTIK

2109116014



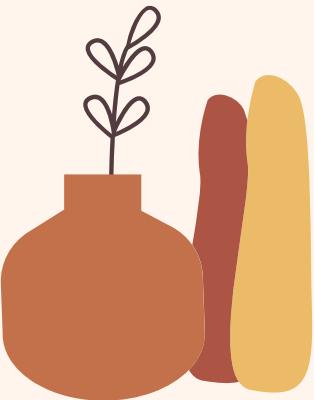
NOVIANA RAMADHANI

2109116006



RAYHAN ABDILAH RAPIQ

2109116022



# Alur CRISP-DM

Business Understanding	Data Understanding	Data Preparation	Modeling	Evaluation	Deployment
<b>Determine Business Objectives</b> <i>Background</i> <i>Business Objectives</i> <i>Business Success Criteria</i>  <b>Assess Situation</b> <i>Inventory of Resources Requirements, Assumptions, and Constraints</i> <i>Risks and Contingencies</i> <i>Terminology</i> <i>Costs and Benefits</i>  <b>Determine Data Mining Goals</b> <i>Data Mining Goals</i> <i>Data Mining Success Criteria</i>  <b>Produce Project Plan</b> <i>Project Plan</i> <i>Initial Assessment of Tools and Techniques</i>	<b>Collect Initial Data</b> <i>Initial Data Collection Report</i>  <b>Describe Data</b> <i>Data Description Report</i>  <b>Explore Data</b> <i>Data Exploration Report</i>  <b>Verify Data Quality</b> <i>Data Quality Report</i>	<b>Select Data</b> <i>Rationale for Inclusion/Exclusion</i>  <b>Clean Data</b> <i>Data Cleaning Report</i>  <b>Construct Data</b> <i>Derived Attributes</i> <i>Generated Records</i>  <b>Integrate Data</b> <i>Merged Data</i>  <b>Format Data</b> <i>Reformatted Data</i>  <b>Dataset</b> <i>Dataset Description</i>	<b>Select Modeling Techniques</b> <i>Modeling Technique</i> <i>Modeling Assumptions</i>  <b>Generate Test Design</b> <i>Test Design</i>  <b>Build Model</b> <i>Parameter Settings</i> <i>Models</i> <i>Model Descriptions</i>  <b>Assess Model</b> <i>Model Assessment</i> <i>Revised Parameter Settings</i>	<b>Evaluate Results</b> <i>Assessment of Data Mining Results w.r.t. Business Success Criteria</i> <i>Approved Models</i>  <b>Review Process</b> <i>Review of Process</i>  <b>Determine Next Steps</b> <i>List of Possible Actions</i> <i>Decision</i>	<b>Plan Deployment</b> <i>Deployment Plan</i>  <b>Plan Monitoring and Maintenance</b> <i>Monitoring and Maintenance Plan</i>  <b>Produce Final Report</b> <i>Final Report</i> <i>Final Presentation</i>  <b>Review Project Experience</b> <i>Documentation</i>

Figure 3: Generic tasks (bold) and outputs (italic) of the CRISP-DM reference model

# Link Collab Dan Data Studio

Collab Supervised :

[https://colab.research.google.com/drive/1mDY19091ioW3Xx8Mr1D\\_o-oRNrEkM6S?usp=sharing](https://colab.research.google.com/drive/1mDY19091ioW3Xx8Mr1D_o-oRNrEkM6S?usp=sharing)

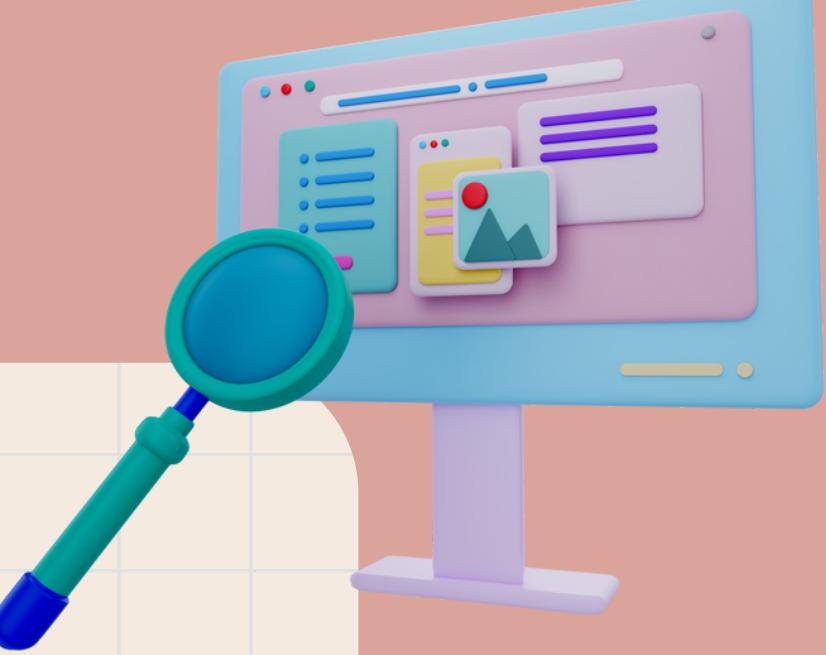
Collab Unsupervised :

<https://colab.research.google.com/drive/1KY29pZ3tQY8-WnDgayBbFe00Wq82Mo2M?usp=sharing>

Data Studio :

<https://lookerstudio.google.com/u/0/navigation/reporting>

# Supervised Dataset

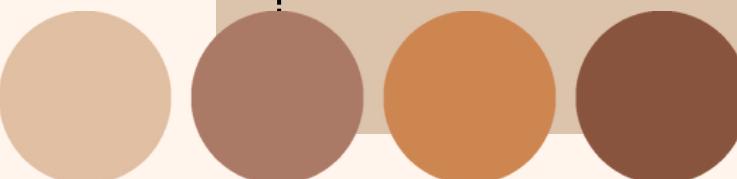


## BUSINESS UNDERSTANDING



Survey Food Student merupakan kumpulan data yang berisi hasil survei tentang kebiasaan makan dan preferensi makanan dari sekelompok siswa.

Tujuan survei adalah untuk mengidentifikasi sikap dan kebiasaan tentang konsumsi makanan di sekolah dan di luar sekolah.

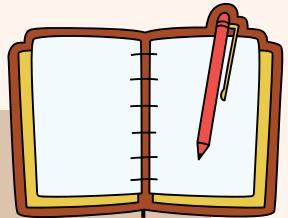


# TUJUAN

Tujuan menggunakan data mining Untuk  
Untuk mencari prediksi tingkat nutrisi dari  
kebiasaan makan siswa berdasarkan  
kebiasaan mereka.

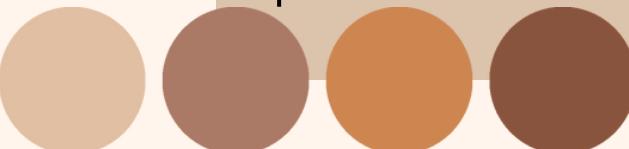


# DATA UNDERSTANDING



## COLLECT DATA

**[https://www.kaggle.com/datasets/mlomuscio  
/student-food-survey](https://www.kaggle.com/datasets/mlomuscio/student-food-survey)**



# Describe Dataset

	<b>Timestamp</b>	<b>Waktu dan tanggal</b>
	<b>Gender</b>	<b>Jenis kelamin</b>
	<b>Asrama</b>	<b>Hari atau Asrama</b>
	<b>Kelas</b>	<b>kelas</b>
	<b>Atlet</b>	<b>Status</b>
	<b>Aktivitas</b>	<b>Aktivitas</b>
	<b>DHBreakfast</b>	<b>Berapa hari dalam seminggu makan sarapan di ruang makan</b>
	<b>NDHBreakfast</b>	<b>Berapa hari dalam seminggu makan sarapan tetapi tidak di ruang makan</b>

# Describe Dataset

BClass	<b>Apakah Anda akan lebih mungkin untuk sarapan jika Anda bisa memakannya di kelas</b>
DHBoxes	<b>Rata-rata, berapa kotak makanan yang Anda makan setiap kali makan di ruang makan</b>
NDHBoxes	<b>Berapa kotak yang Anda ambil dari ruang makan untuk dimakan nanti</b>
NDHMeals	<b>Berapa kali seminggu Anda makan di asrama atau pusat siswa</b>
Nutrisi	<b>Pada skala 0 sampai 5, seberapa sadarkah Anda tentang nilai gizi dari makanan yang Anda makan</b>
Uang	<b>Rata-rata, berapa banyak uang yang dibelanjakan untuk makanan di luar ruang makan per minggu</b>

# Membaca Dataset

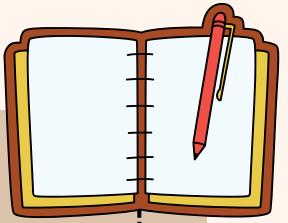
## ▼ Data Understanding

Tahap ini melibatkan memahami dataset yang akan digunakan, melihat fitur apa yang tersedia dan mempelajari apakah ada data yang hilang atau tidak lengkap. Pada tahap ini, kita perlu mempelajari dataset student-food-survey dan mencari tahu apa yang tersedia dalam dataset tersebut dan apakah ada data yang hilang atau tidak lengkap.

```
✓ [1] 1 # import library
      2 import pandas as pd
      3 import numpy as np
      4 import math

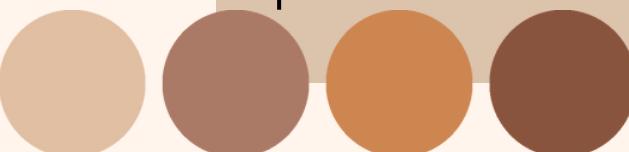
✓ [2] 1 food_survey= "https://drive.google.com/file/d/1orRhv3rtUm_ekzvNDIi2h8srz4ZZSof0/view?usp=share_link"
      2 data_food_survey= "https://drive.google.com/uc?id=1orRhv3rtUm_ekzvNDIi2h8srz4ZZSof0"

✓ [3] 1 # Membaca dataset
      2 df = pd.read_csv(data_food_survey)
      3 df
```

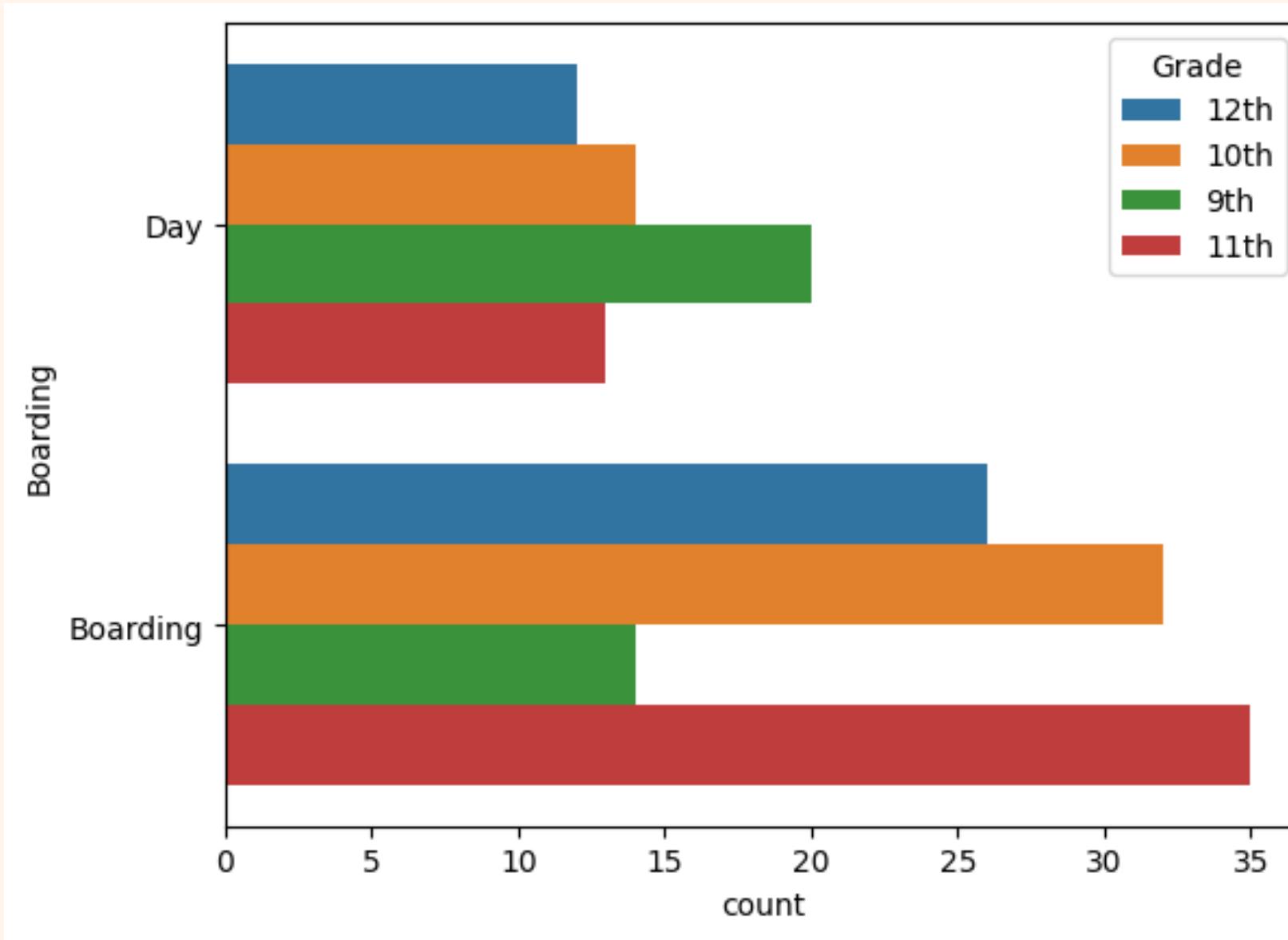


# Explorating Data

Suatu proses analisis untuk memahami, meringkas, dan menggambarkan karakteristik dasar dari suatu set data secara visual dan kuantitatif.

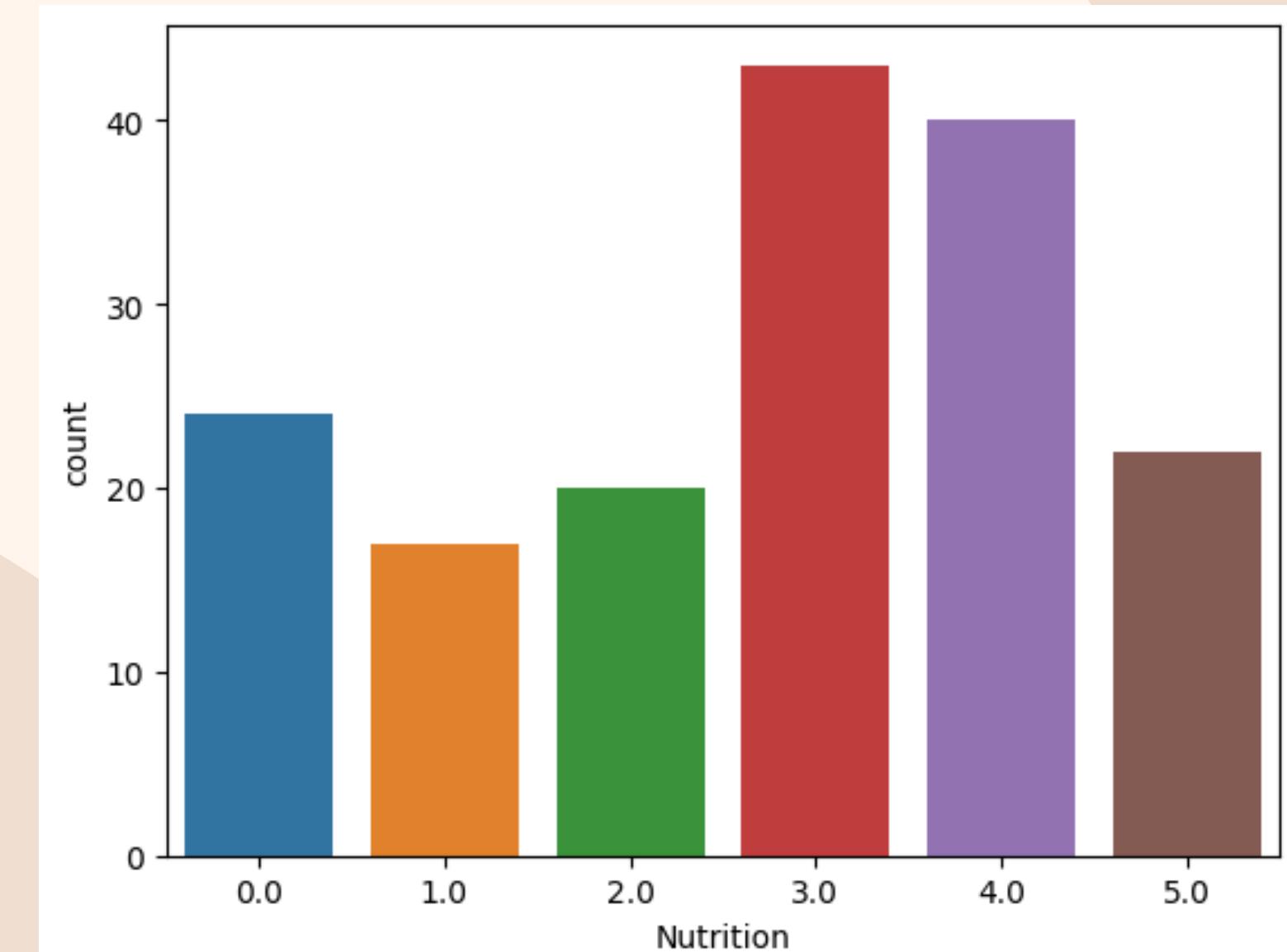


# Visualisasi Comparison

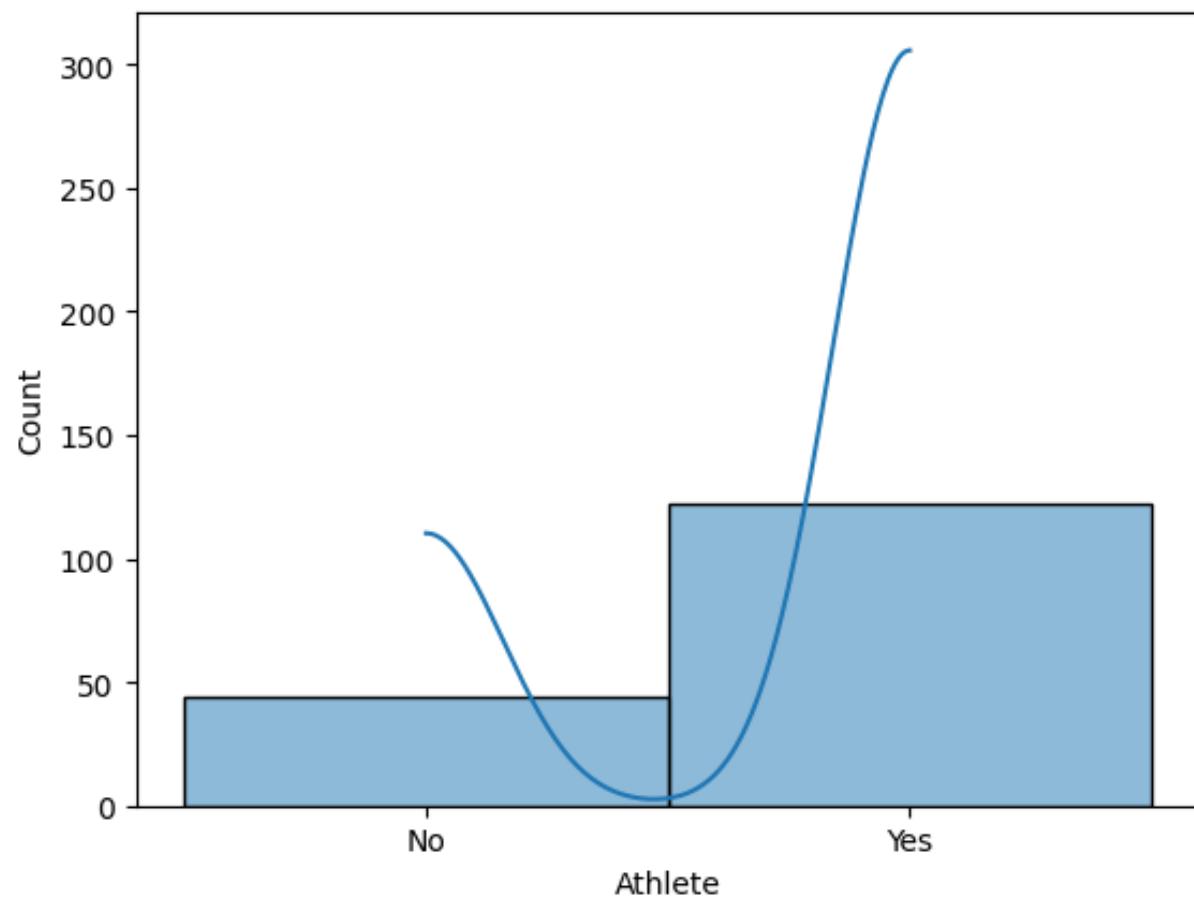


Menunjukkan jumlah siswa yang tinggal di asrama, sedangkan setiap warna menunjukkan jumlah siswa di setiap tingkat kelas.

Menampilkan jumlah data pada setiap kategori dalam kolom "Nutrition" dari dataset

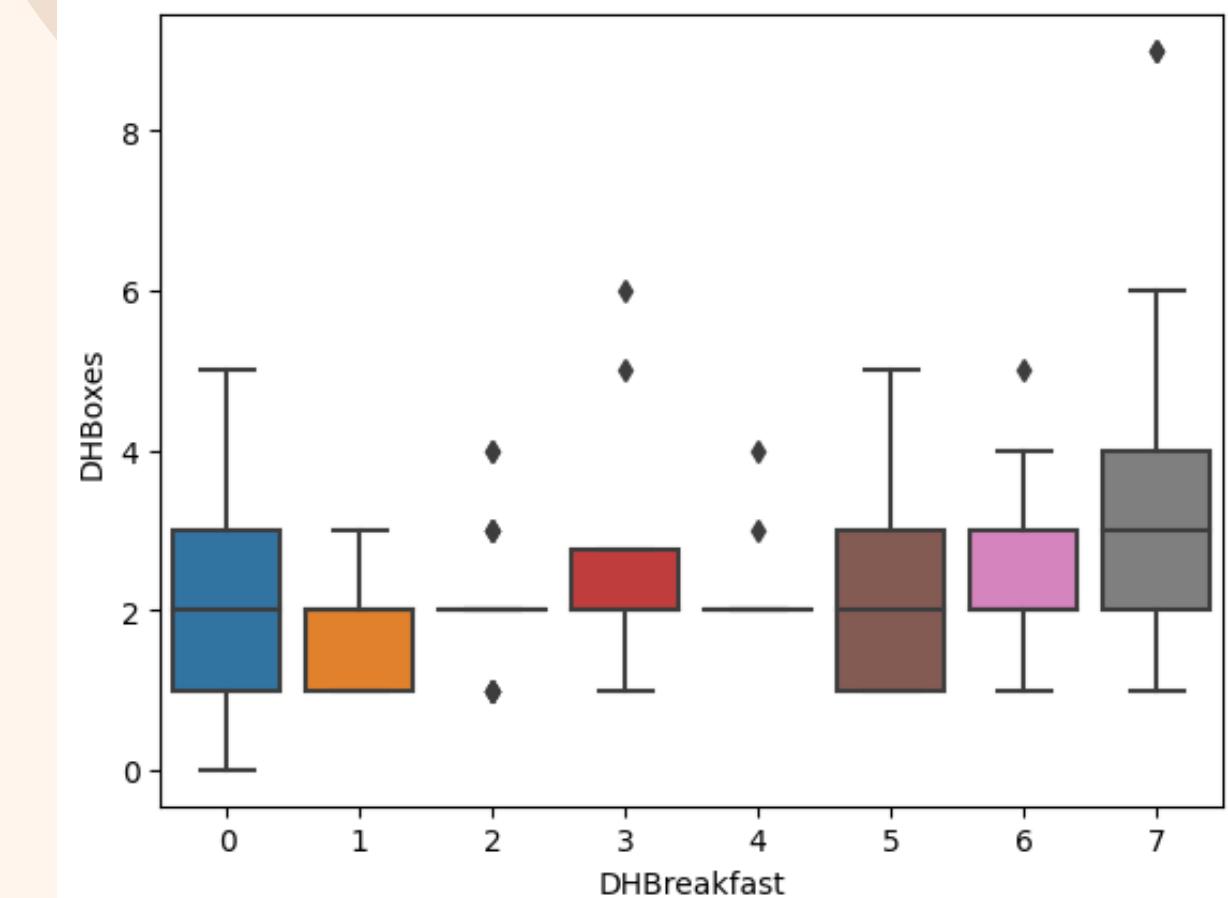
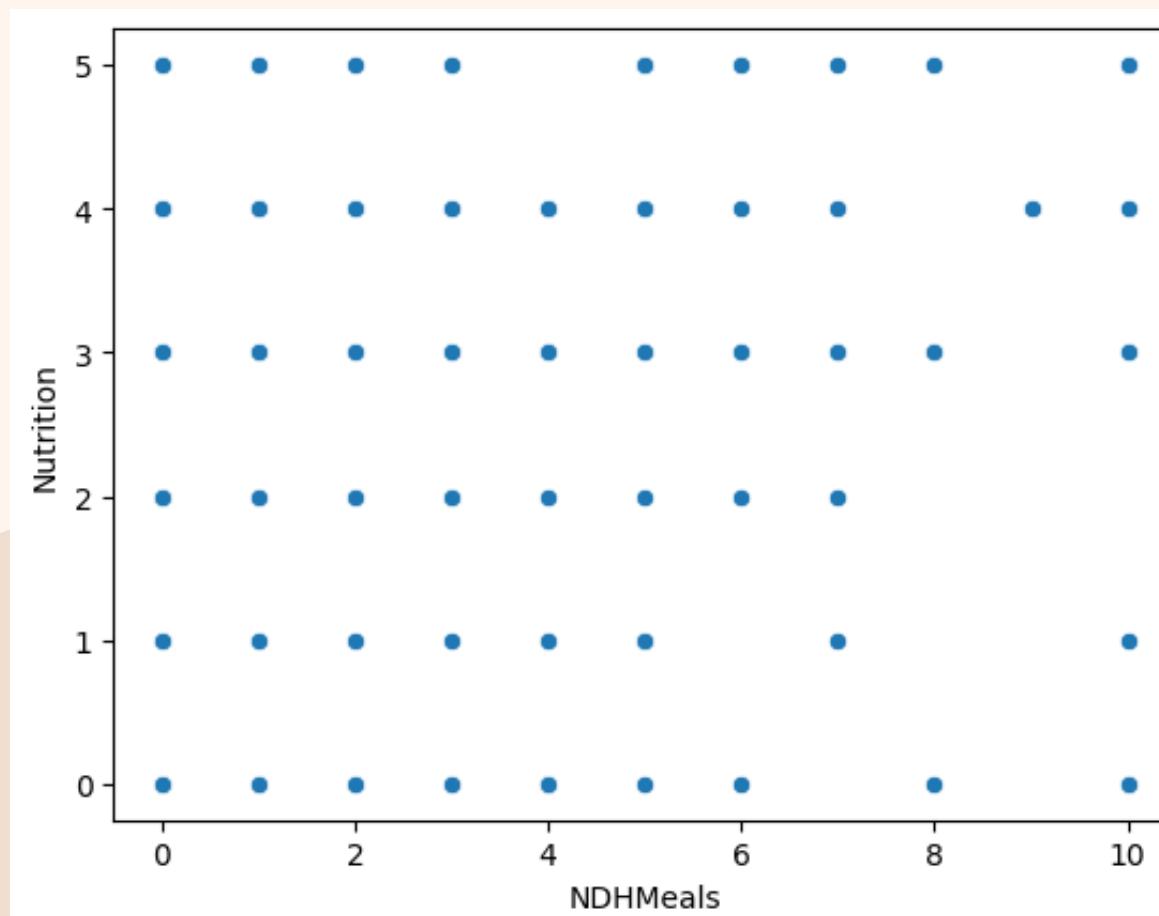


# Visualisasi Distribution



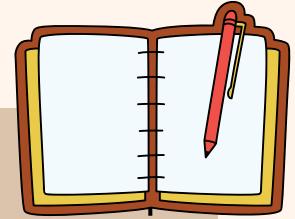
histogram (grafik distribusi frekuensi) dari kolom "Athlete"

visualisasi hubungan antara konsumsi makanan tidak di asrama (Non-Boarding Meals) dan tingkat Nutrisi pada siswa.



menampilkan kotak berisi sebagian data, dengan garis tengah menunjukkan nilai median, batas bawah menunjukkan kuartil bawah dan batas atas menunjukkan kuartil atas, serta tanda-tanda bintang menunjukkan nilai-nilai pencilan.

# Data Preparation



**Tahap ini untuk mempersiapkan data untuk analisis, termasuk membersihkan data yang tidak relevan, memperbaiki data yang rusak, dan membuat dataset siap digunakan.**



# Cleansing Data

menghitung jumlah  
nilai null (kosong)  
dalam setiap kolom

```
[14] # Cek missing values
      print(dnew.isnull().sum())

      DHBreakfast          0
      NDHBreakfast         0
      DHBoxes              2
      NDHBoxes             3
      NDHMeals             0
      Nutrition            1
      Money                26
      Gender_Female        0
      Gender_Male          0
      Gender_Non-Binary    0
      Gender_Other          0
      Boarding_Boarding     0
      Boarding_Day          0
      Grade_10th            0
      Grade_11th            0
      Grade_12th            0
      Grade_9th             0
      Activities_Cross Country 0
      Activities_Football   0
      Activities_Golf        0
      Activities_Intramurals 0
      Activities_Musical    0
      Activities_None of the above 0
      Activities_Pre-season conditioning 0
      Activities_Soccer     0
      Activities_Tennis      0
      Activities_Volleyball 0
      BClass_No              0
      BClass_Yes             0
      dtype: int64
```

menghapus semua  
baris yang maemiliki  
nilai null (missing value)

```
[15] #Menghapus baris Data Kosong / prepocessing data
      dnew = dnew.dropna()
      dnew.isna().sum()
```

menghapus kolom  
'Timestamp' dan 'Athlete'  
karena tidak relevan dan  
melihat data duplikat

```
[11] #Menghapus kolom yang tidak relevan
      dnew = df.drop(['Timestamp', 'Athlete'], axis=1)
      dnew
```

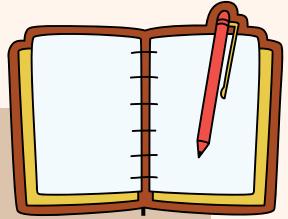
```
[13] #Melihat data duplikat
      duplikat =dnew.duplicated()
      print(dnew[duplikat])
```

# Membagi Data Latih Dan Data Uji

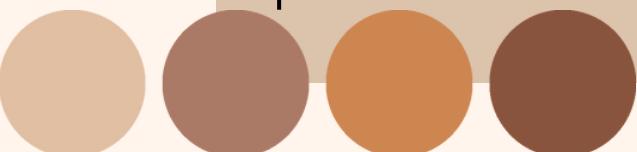
```
[17] from sklearn.model_selection import train_test_split  
x = dnew.drop('Nutrition',axis=1)  
y = dnew['Nutrition'] #kolom target  
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)  
print("Data Latih Yang Akan Digunakan :\n",x_train)  
print("Data Uji Yang Akan Digunakan :\n",x_test)
```

contoh implementasi dari pustaka Scikit-learn (sklearn) untuk membagi data menjadi data latih (train) dan data uji (test) menggunakan fungsi `train_test_split`.

## Modelling - I



# Naive Bayes



# Naïve Bayes

```
def naive_bayes_crisp_dm(dataset, target_column):
    # Menghitung jumlah kelas dan jumlah nilai pada setiap atribut
    class_values, attribute_values = {}, {}
    for col in dataset.columns:
        if col != target_column:
            attribute_values[col] = list(set(dataset[col]))
    class_counts = dataset[target_column].value_counts()

    # Menghitung probabilitas prior untuk setiap kelas
    class_probs = {}
    for cls in class_counts.index:
        class_probs[cls] = class_counts[cls] / len(dataset)

    # Menghitung probabilitas kondisional untuk setiap nilai atribut pada setiap kelas
    conditional_probs = {}
    for col in attribute_values.keys():
        for cls in class_counts.index:
            class_data = dataset[dataset[target_column] == cls]
            class_size = len(class_data)
            for val in attribute_values[col]:
                val_data = class_data[class_data[col] == val]
                val_size = len(val_data)
                prob = (val_size + 1) / (class_size + len(attribute_values[col]))
                key = (col, val, cls)
                conditional_probs[key] = prob
```

1. menghitung jumlah kelas dan jumlah nilai setiap atribut
2. menghitung probabilitas prior setiap kelas
3. menghitung probabilitas setiap nilai atribut pada setiap kelas

# Naïve Bayes

```
# Melakukan prediksi pada dataset
correct_predictions = 0
for index, row in dataset.iterrows():
    max_prob, max_class = -1, None
    for cls in class_counts.index:
        prob = class_probs[cls]
        for col in attribute_values.keys():
            key = (col, row[col], cls)
            if key in conditional_probs:
                prob *= conditional_probs[key]
        if prob > max_prob:
            max_prob, max_class = prob, cls
    if max_class == row[target_column]:
        correct_predictions += 1

# Menghitung akurasi prediksi
accuracy = correct_predictions / len(dataset)

return accuracy
```

4. Melakukan prediksi
5. Menghitung akurasi

# Hasil prediksi



Dataset shape: (167, 14)

First 5 rows:

	Timestamp	Gender	Boarding	Grade	Athlete	Activities	\
0	10/22/2020 11:36:55	Female		Day	12th	No	None of the above
1	10/22/2020 11:36:58	Male	Boarding	10th	Yes		Soccer
2	10/22/2020 11:36:59	Female		Day	9th	Yes	Tennis
3	10/22/2020 11:37:00	Male		Day	11th	No	None of the above
4	10/22/2020 11:37:01	Female		Day	12th	No	Musical

	DHBreakfast	NDHBreakfast	BClass	DHBoxes	NDHBoxes	NDHMeals	Nutrition	\
0	0	2	No	2.0	0.0	0	0.0	
1	5	2	No	2.0	0.0	5	4.0	
2	3	7	Yes	2.0	0.0	10	4.0	
3	0	4	No	2.0	0.0	1	2.0	
4	0	5	Yes	1.0	0.0	2	3.0	

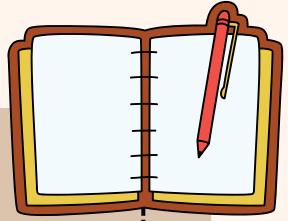
	Money
0	40.0
1	15.0
2	50.0
3	4.0
4	20.0

Accuracy: 0.7485029940119761

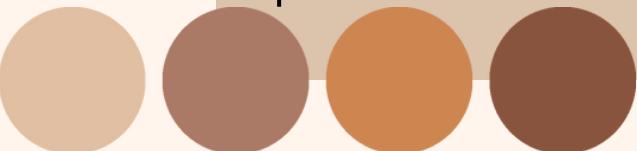
## Kesimpulan prediksi naive bayes

Output "Accuracy: 0.7485029940119761" menunjukkan bahwa model Naive Bayes yang dijalankan pada dataset memiliki akurasi sebesar 0.7485 atau sekitar 74.85%. Akurasi ini menunjukkan seberapa baik model dapat memprediksi kelas target (dalam hal ini, kategori "Nutrition" pada dataset)

## Modelling - II



# KNN



# Alur KNN Modelling

```
import pandas as pd
import numpy as np
import math

# Menghapus kolom dengan tipe data string
dnew = dnew.select_dtypes(exclude=['object'])

# Menentukan fitur dan target
X = dnew.drop('Nutrition', axis=1).values.astype(float)
y = dnew['Nutrition'].values

# Mengimputasi nilai yang hilang
X = np.nan_to_num(X, nan=np.nanmedian(X, axis=0))
y = pd.Series(y).fillna(pd.Series(y).mode()[0]).values

# Membagi data menjadi set pelatihan dan pengujian
np.random.seed(42)
indices = np.random.permutation(len(X))
train_size = int(0.8 * len(X))
X_train = X[indices[:train_size]]
y_train = y[indices[:train_size]]
X_test = X[indices[train_size:]]
y_test = y[indices[train_size:]]

# Melatih klasifikasi KNN dengan k=5
def euclidean_distance(x1, x2):
    distance = 0
    for i in range(len(x1)):
        distance += (x1[i] - x2[i]) ** 2
    return math.sqrt(distance)
```

- Menghapus kolom dengan tipe data string dari sebuah dataframe yang disimpan dalam variabel 'dnew'.
- Menentukan fitur dan target dari dataset 'dnew'.
- Mengimputasi nilai yang hilang dalam fitur dan target dengan nilai median untuk fitur dan nilai modus untuk target.
- Membagi data menjadi set pelatihan dan pengujian dengan perbandingan 80:20.
- Melatih klasifikasi KNN dengan fungsi jarak Euclidean dan nilai k=5.



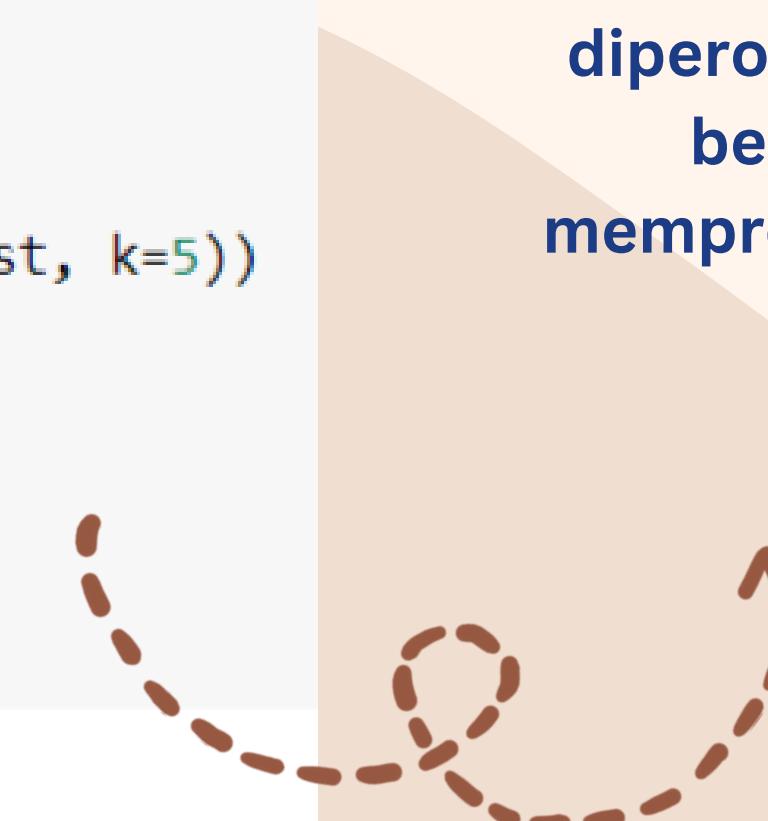
# Alur KNN Modelling

```
def predict(x_train, y_train, x_test, k):
    distances = []
    for i in range(len(x_train)):
        distance = euclidean_distance(x_train[i], x_test)
        distances.append((distance, y_train[i]))
    distances.sort()
    neighbors = [distances[i][1] for i in range(k)]
    return max(set(neighbors), key=neighbors.count)

y_pred = []
for x_test in x_test:
    y_pred.append(predict(x_train, y_train, x_test, k=5))

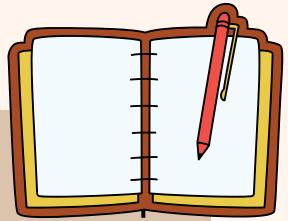
# Menghitung akurasi model
accuracy = sum(y_pred == y_test) / len(y_test)
print(f'Akurasi: {accuracy.item():.2f}')
```

Akurasi: 0.21

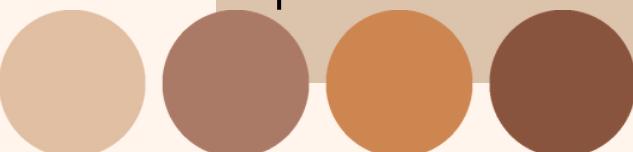


Output Akurasi: 0.21 menunjukkan akurasi model klasifikasi KNN dengan k=5 pada dataset "Food Survey.csv" untuk memprediksi kategori "Nutrition". Dalam hal ini, nilai akurasi yang diperoleh adalah sebesar 0.22 atau 22%, yang berarti model ini kurang efektif untuk memprediksi kategori "Nutrition" pada dataset tersebut.

## Modelling - III



# C4.5



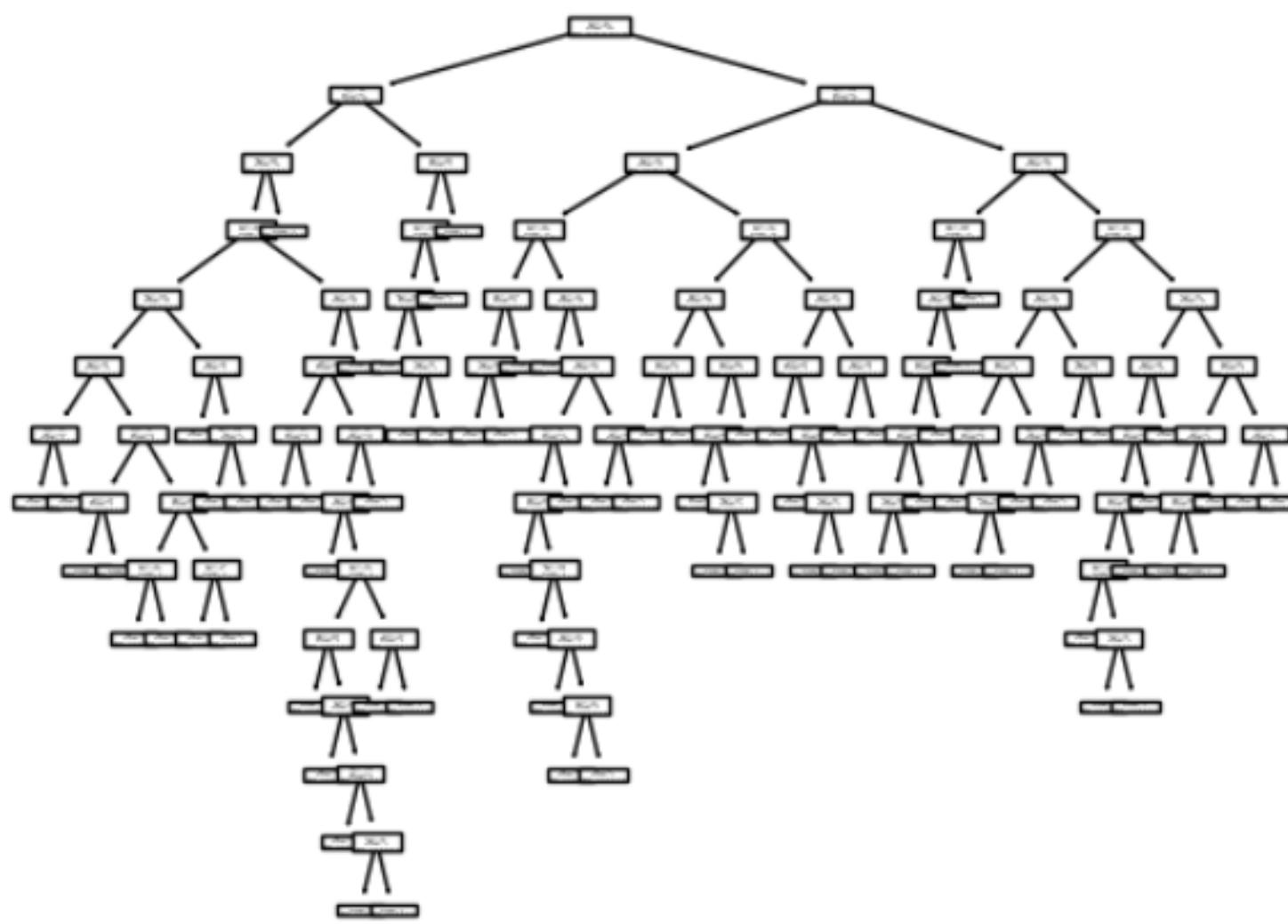
# C4.5

```
1 # import library
2 import pandas as pd
3 from sklearn.tree import DecisionTreeClassifier
4 from sklearn.model_selection import train_test_split
5 from sklearn.metrics import accuracy_score
6 from sklearn.preprocessing import LabelEncoder, OneHotEncoder
7
8 # Melakukan One-Hot Encoding pada kolom "Nutrition"
9 ohe = OneHotEncoder()
10 Nutrition = ohe.fit_transform(df[['Nutrition']])
11 df_Nutrition = pd.DataFrame(Nutrition.toarray(), columns=ohe.get_feature_names_out(['Nutrition']))
12 df = pd.concat([df, df_Nutrition], axis=1)
13
14 # Membuat model dengan algoritma C4.5
15 model = DecisionTreeClassifier(criterion='entropy', max_depth=3)
16
17 # Melatih model dengan data latih
18 model.fit(X_train, y_train)
19
20 # Memprediksi kategori "Nutrition" pada data uji
21 y_pred = model.predict(X_test)
22 print(y_pred[0:5])
23 print(y_test[0:5])
24
```

# C4.5

```
25 from sklearn.metrics import accuracy_score,ConfusionMatrixDisplay,roc_auc_score,confusion_matrix  
26 from sklearn.model_selection import KFold,StratifiedKFold,RandomizedSearchCV  
27 from sklearn.metrics import roc_curve, auc  
28 from sklearn.tree import DecisionTreeClassifier,plot_tree  
29 c45 = DecisionTreeClassifier()  
30 c45.fit(X_train,y_train)  
31 pred_c45 = c45.predict(X_test)  
32 plot_tree(c45)  
33 cm= confusion_matrix(y_test, pred_c45)
```

[3. 2. 3. 3. 3.]  
[0. 3. 3. 1. 5.]



```
1 # Menghitung akurasi model  
2 accuracy = accuracy_score(y_test, y_pred)  
3 print("Akurasi model:", accuracy)
```

Akurasi model: 0.14705882352941177

Diperoleh akurasi model sebesar 0.14705. Meskipun cukup tinggi, namun masih terdapat ruang untuk perbaikan dan peningkatan performa model dengan mengoptimalkan hyperparameter, melakukan pemilihan fitur, dan memperbaiki kualitas data.

# Evaluation



Pada tahap ini, dilakukan evaluasi terhadap model atau solusi yang telah dibuat untuk memastikan apakah sesuai dengan kebutuhan bisnis atau tidak. Evaluasi dilakukan dengan melihat beberapa kriteria evaluasi, seperti akurasi, kecepatan, efisiensi, dan fitur-fitur lainnya.



# Naive Bayes

```
[ ] # Evaluasi Naive Bayes
```

```
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score, roc_auc_score
from sklearn.naive_bayes import GaussianNB, MultinomialNB
gnb = GaussianNB()
y_pred = gnb.fit(x_train, y_train).predict(x_test)

accuracy_bys = accuracy_score(y_test, y_pred)
print("Akurasi: {:.2f}%".format(accuracy_bys*100))

precision_bys = precision_score(y_test, y_pred, average="macro")
print("Presisi: {:.2f}%".format(precision_bys*100))

recall_bys = recall_score(y_test, y_pred, average="macro")
print("Recall: {:.2f}%".format(recall_bys*100))

f1_bys = f1_score(y_test, y_pred, average="macro")
print("F-1: {:.2f}%".format(f1_bys*100))

Y_gnb_score = gnb.predict_proba(x_test)
lr = LogisticRegression()
lr.fit(x_train, y_train)
roc_auc_bys = roc_auc_score(y_test,Y_gnb_score, multi_class='ovr')
print("AUC ROC: {:.2f}%".format(roc_auc_bys*100))
```

Akurasi: 14.29%  
Presisi: 5.44%  
Recall: 18.33%  
F-1: 8.35%  
AUC ROC: 41.36%

# KNN

```
1 import numpy as np
2 import seaborn as sns
3 from sklearn.tree import DecisionTreeClassifier
4 from sklearn.metrics import roc_auc_score, precision_score, recall_score, f1_score, confusion_matrix
5
6 # definisikan model Decision Tree Classifier dengan beberapa parameter
7 clf = DecisionTreeClassifier(max_depth=3, random_state=42)
8
9 # train model dengan data training
10 clf.fit(X_train, y_train)
11
12 # prediksi probabilitas target variabel pada data test
13 y_pred_prob = clf.predict_proba(X_test)
14
15 # hitung AUC
16 roc_auc_knn = roc_auc_score(y_test, y_pred_prob, multi_class='ovr')
17 print(f'AUC: {roc_auc_knn:.2f}')
18
19 # prediksi target variabel pada data test
20 y_pred = clf.predict(X_test)
21
22 #Menghitung akurasi model
23 accuracy_knn = sum(y_pred == y_test) / len(y_test)
24 print(f'Akurasi: {accuracy_knn:.2f}')
25
26 # hitung precision
27 precision_knn = precision_score(y_test, y_pred, average='macro', zero_division=0)
28 print(f'Precision: {precision_knn:.2f}')
29
30 # hitung recall
31 recall_knn = recall_score(y_test, y_pred, average='macro', zero_division=0)
32 print(f'Recall: {recall_knn:.2f}')
33
```

# KNN

```
19 # prediksi target variabel pada data test
20 y_pred = clf.predict(X_test)
21
22 #Menghitung akurasi model
23 accuracy_knn = sum(y_pred == y_test) / len(y_test)
24 print(f'Akurasi: {accuracy_knn:.2f}')
25
26 # hitung precision
27 precision_knn = precision_score(y_test, y_pred, average='macro', zero_division=0)
28 print(f'Precision: {precision_knn:.2f}')
29
30 # hitung recall
31 recall_knn = recall_score(y_test, y_pred, average='macro', zero_division=0)
32 print(f'Recall: {recall_knn:.2f}')
33
34 # hitung F1 Score
35 f1_knn = f1_score(y_test, y_pred, average='macro', zero_division=0)
36 print(f'F1 Score: {f1_knn:.2f}')
37
38 # membuat confusion matrix
39 cm = confusion_matrix(y_test, y_pred)
40
41 #membuat heatmap confusion matrix
42 sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
43
44 #tambahkan judul dan label pada plot
45 plt.title('Confusion Matrix')
46 plt.xlabel('Predicted Label')
47 plt.ylabel('True Label')
48
49 #tampilkan plot
50 plt.show()
```

# HASIL KNN

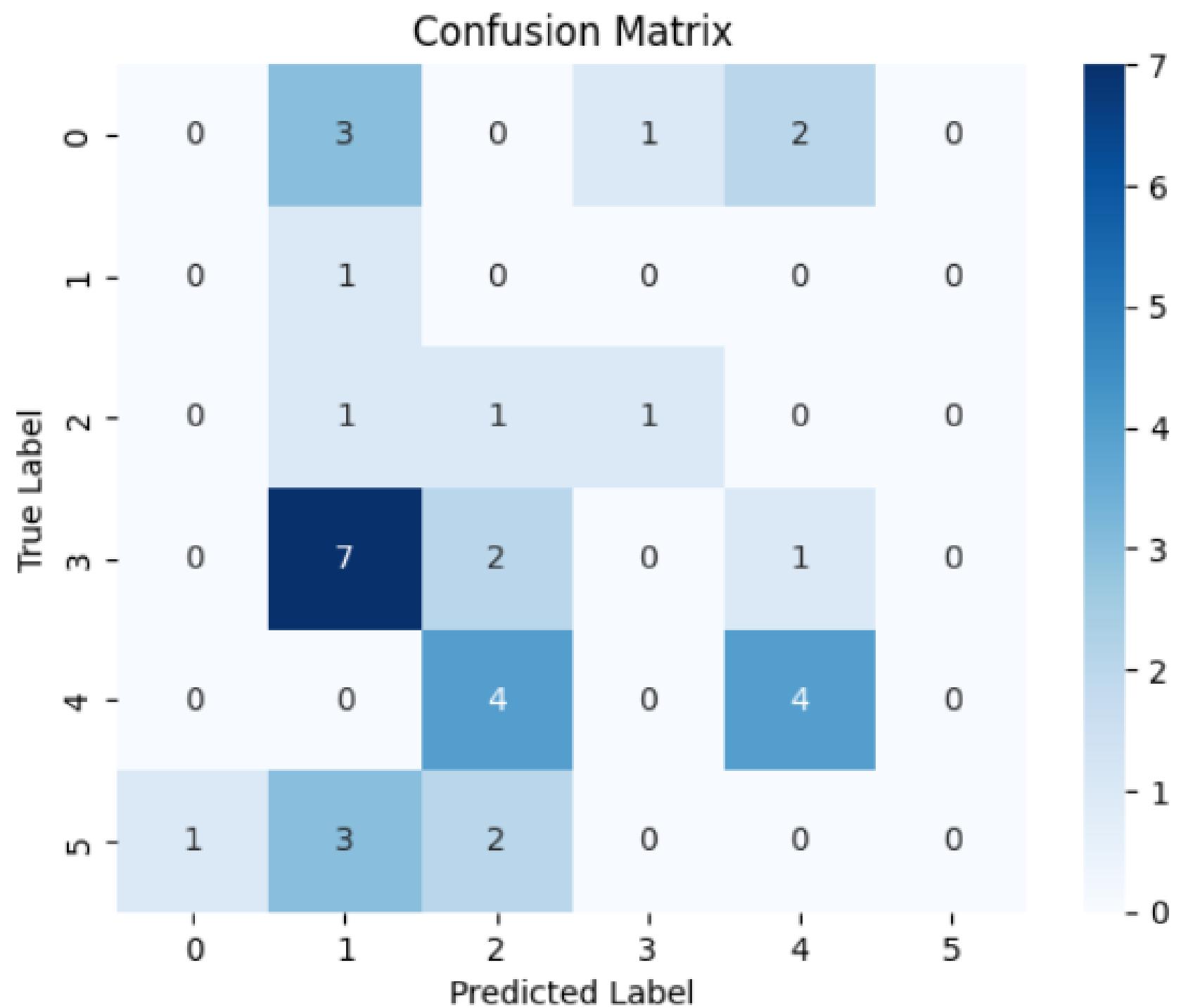
AUC: 0.67

Akurasi: 0.18

Precision: 0.12

Recall: 0.31

F1 Score: 0.14



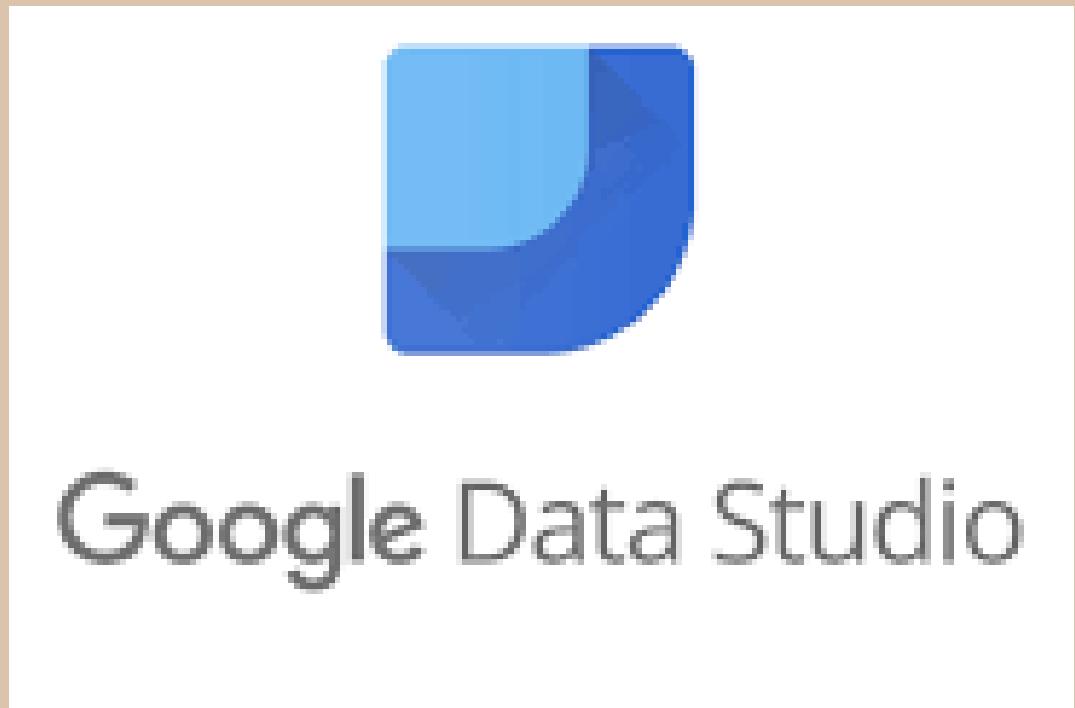
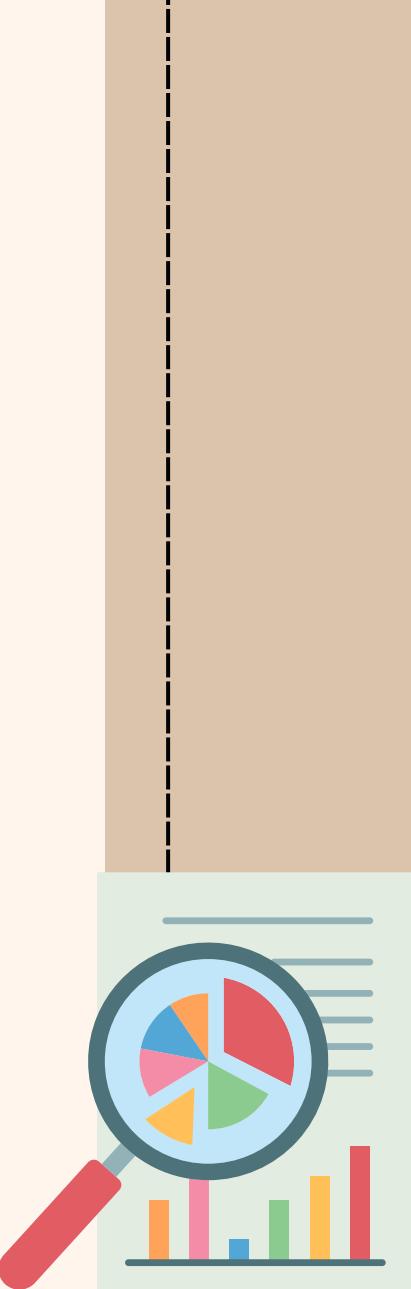
# C4.5

```
1 # Evaluasi C4.5
2
3 from sklearn.metrics import accuracy_score, classification_report, roc_auc_score, precision_score
4 from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
5 # Melakukan prediksi dengan data testing
6 y_pred = model.predict(X_test)
7
8 # Akurasi
9 accuracy = accuracy_score(y_test, y_pred)
10 print("Akurasi: {:.2f}%".format(accuracy*100))
11
12 # Presisi
13 precision = precision_score(y_test, y_pred, average='macro')
14 print("Presisi: {:.2f}%".format(precision*100))
15
16 # Recall
17 recall = recall_score(y_test, y_pred, average='macro')
18 print("Recall: {:.2f}%".format(recall*100))
19
20 # F-1
21 f1 = f1_score(y_test, y_pred, average='macro')
22 print("F-1: {:.2f}%".format(f1*100))
23
24 # ROC AUC
25 from sklearn.linear_model import LogisticRegression
26 clf = LogisticRegression(solver="liblinear").fit(X_train, y_train)
27 preds = clf.predict_proba(X_test)
28 roc_auc = roc_auc_score(y_test,preds, multi_class='ovr')
29 print("ROC AUC: {:.2f}%".format(roc_auc*100))
```



Akurasi: 14.71%  
Presisi: 3.79%  
Recall: 8.33%  
F-1: 5.21%  
ROC AUC: 62.19%

# Deployment



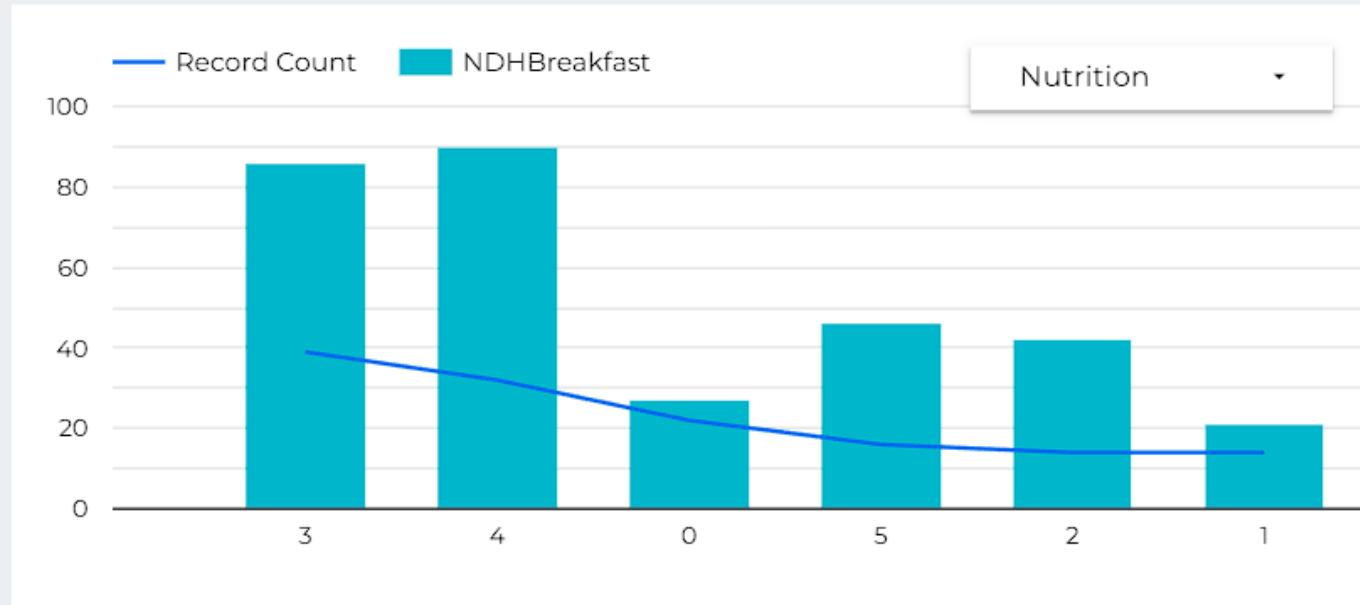
# Supervised

## FOOD survey Student

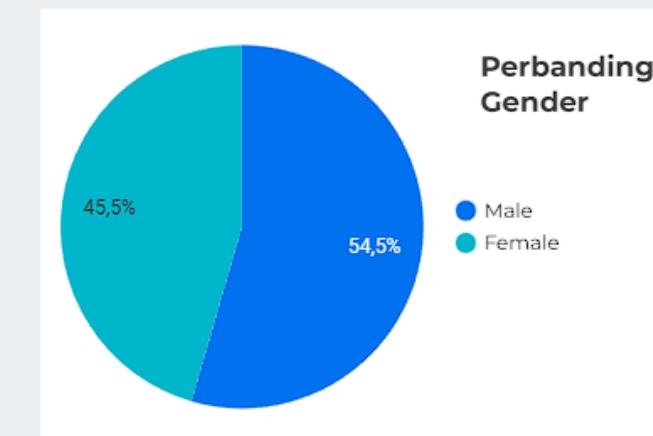
DHBoxes  
311,0

Activities  
10

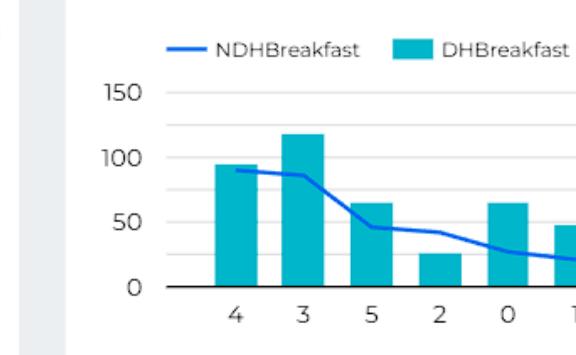
Survey Food Student adalah kumpulan data yang berisi hasil survei tentang kebiasaan makan dan preferensi makanan dari sekelompok siswa. Tujuan survei adalah untuk mengidentifikasi sikap dan kebiasaan tentang konsumsi makanan di sekolah dan di luar sekolah.



## VISUALISASI



Grafik Nutrisi Sarapan disekolah dan diluar sekolah

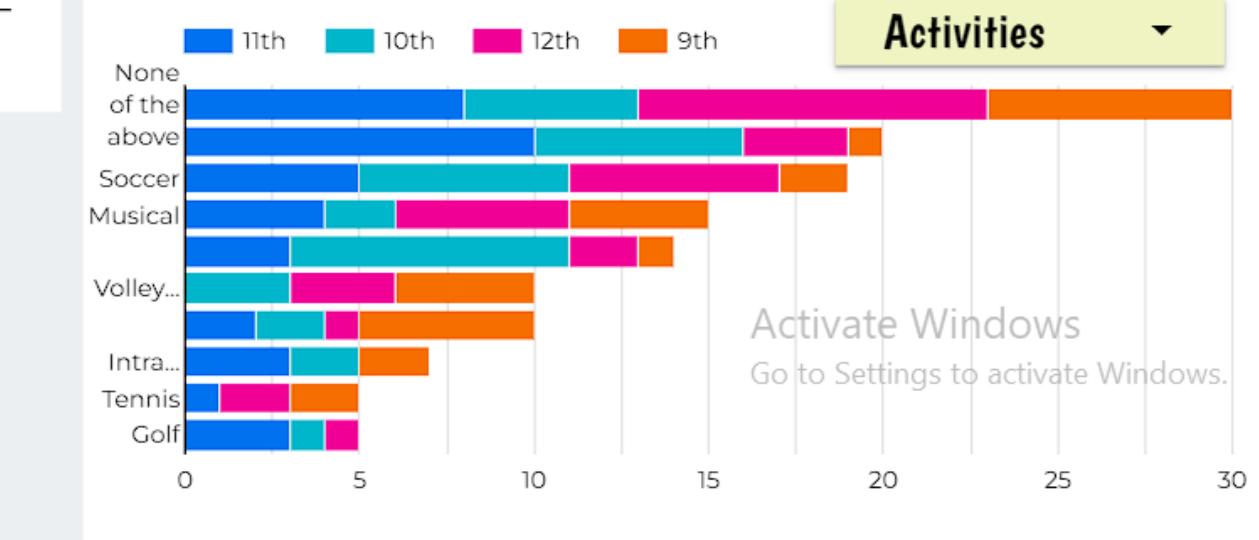
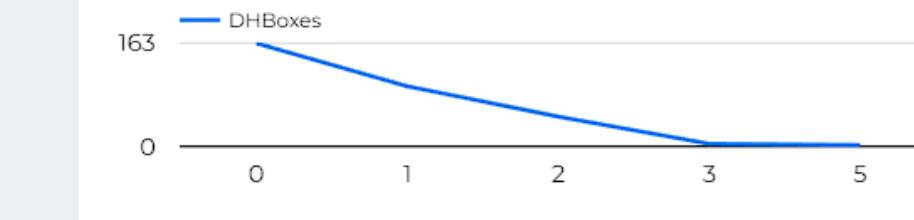


Grafik siswa makan di ruang makan berdasarkan status siswa

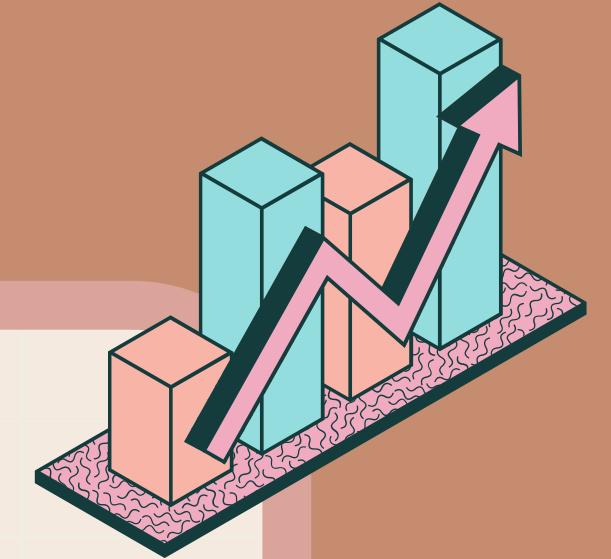
Nutrition	NDHBreakfast	DHBreakfast	Money
1.	4	90	96
2.	3	86	119
3.	5	46	65
4.	2	42	26
5.	0	27	66
6.	1	21	48

1 - 6 / 6 < >

Grafik banyaknya kotak makan yang diambil



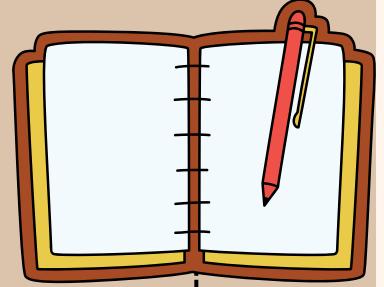
Activate Windows  
Go to Settings to activate Windows.



# Unsupervised Dataset



# BUSINESS UNDERSTANDING

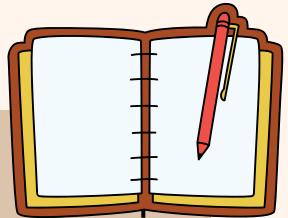


Dataset "Countries of the World" dari Kaggle adalah kumpulan data yang berisi informasi mengenai berbagai negara di seluruh dunia. Dataset ini memiliki 20 kolom yang memberikan berbagai informasi. Tujuan dari dataset ini adalah untuk membantu para analis dalam mempelajari dan memahami perbedaan antara negara-negara di seluruh dunia dan faktor-faktor yang memengaruhi kesejahteraan penduduk di negara-negara tersebut

Tujuan menggunakan Data Mining  
Mengelompokkan negara-negara  
berdasarkan kepadatan penduduk mereka,  
seperti negara dengan kepadatan penduduk  
tinggi, sedang, atau rendah.

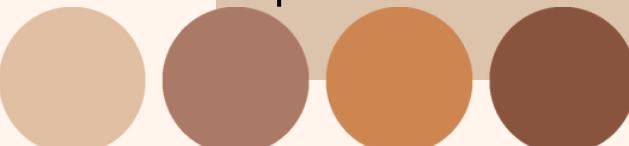


# DATA UNDERSTANDING



## COLLECT DATA

**[https://www.kaggle.com/datasets/fernando/  
countries-of-the-world](https://www.kaggle.com/datasets/fernando/countries-of-the-world)**



# Describe Dataset

<b>Country</b>	<b>Nama Negara</b>
<b>Region</b>	<b>Wilayah geografis di mana negara tersebut berada.</b>
<b>Population</b>	<b>Jumlah penduduk negara.</b>
<b>Area (sq. mi.)</b>	<b>Luas wilayah negara dalam mil persegi.</b>
<b>Pop. Density (per sq. mi.)</b>	<b>Kepadatan penduduk dalam mil persegi.</b>
<b>Coastline (coast/area ratio)</b>	<b>Rasio garis pantai terhadap luas wilayah negara.</b>
<b>Net migration</b>	<b>Angka migrasi neto (jumlah orang yang pindah masuk dikurangi jumlah orang yang pindah keluar) dalam negara tersebut.</b>
<b>Infant mortality (per 1000 births)</b>	<b>Angka kematian bayi per 1000 kelahiran hidup.</b>

# Describe Dataset

GDP (\$ per capita):	Produk domestik bruto (PDB) per kapita dalam dolar AS.
Literacy (%)	Persentase populasi yang bisa membaca dan menulis
Phones (per 1000)	Jumlah telepon seluler per 1000 orang.
Arable (%)	Persentase lahan yang dapat digunakan untuk pertanian.
Crops (%)	Persentase lahan yang ditanami dengan tanaman.
Other (%)	Persentase lahan yang tidak dapat digunakan untuk pertanian.
Climate	Klasifikasi iklim berdasarkan skala Köppen, dengan nilai-nilai yang meliputi: 1 (tropis), 2 (subtropis), 3 (sedang), 4 (dingin), 5 (tundra), dan 6 (gurun).
Birthrate	Jumlah kelahiran per 1000 penduduk dalam setahun

# Describe Dataset

Deathrate	Jumlah kematian per 1000 penduduk dalam setahun.
Agriculture	Persentase PDB yang berasal dari sektor pertanian.
Industry	Persentase PDB yang berasal dari sektor industri
Service	Persentase PDB yang berasal dari sektor jasa.

# Membaca Dataset

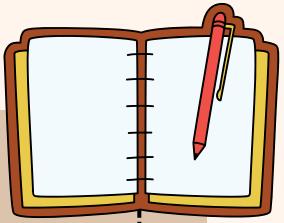
## Data Understanding

```
[1] 1 #Import Library  
2 import pandas as pd  
3 from sklearn.preprocessing import StandardScaler  
  
[3] 1 countries = "https://drive.google.com/file/d/1ljcGvIC9h4WHs2daba2XLRpmmMjmbp8/view?usp=drivesdk"  
2 data_countries = "https://drive.google.com/uc?id=1ljcGvIC9h4WHs2daba2XLRpmmMjmbp8"  
  
# Membaca dataset  
df = pd.read_csv(data_countries)  
df
```

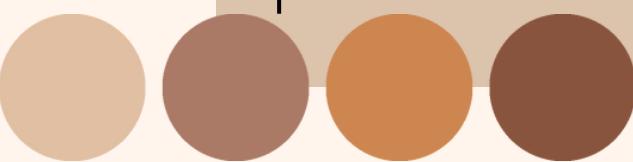
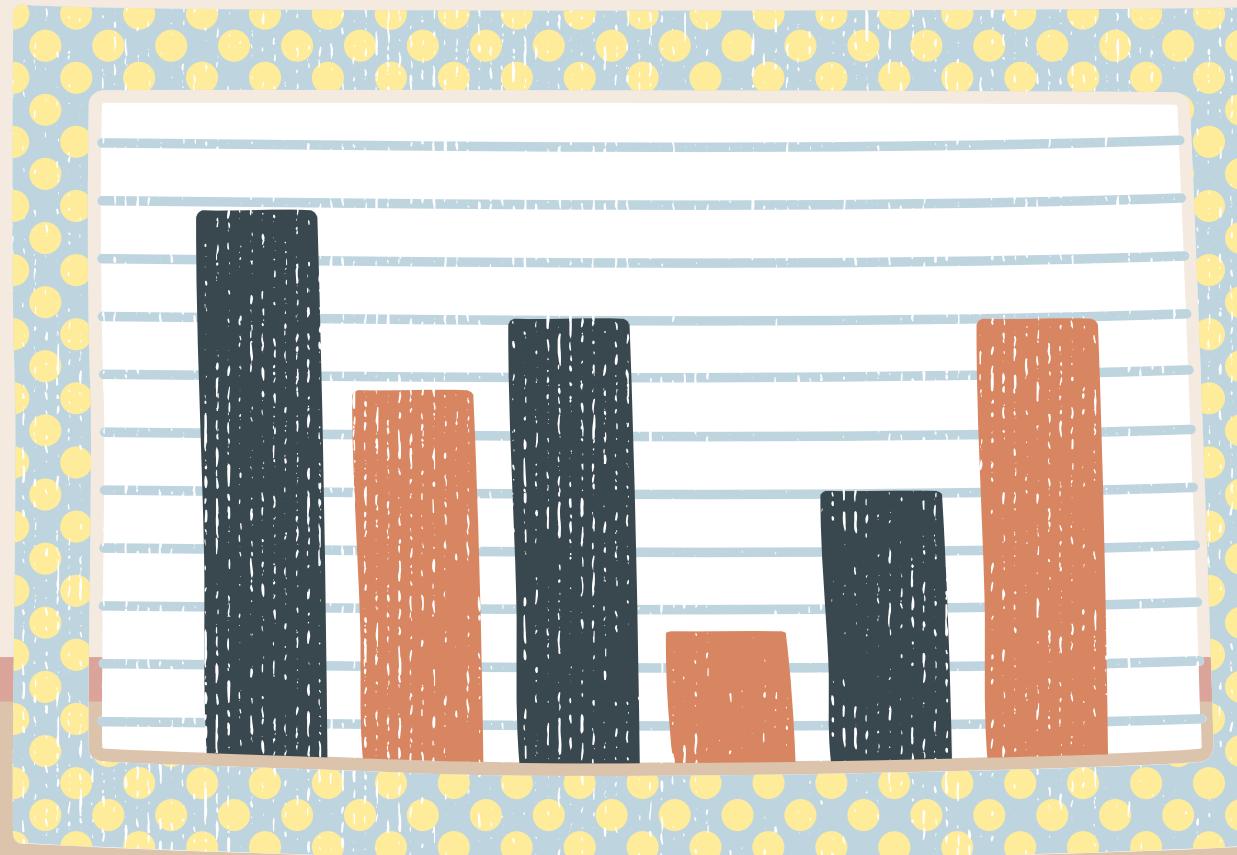
## Mengambil 20% Data secara acak

```
# Mengambil 20% data secara acak  
df_sample = df.sample(frac=0.3, random_state=42)  
# Menampilkan data sample  
print(df_sample)
```

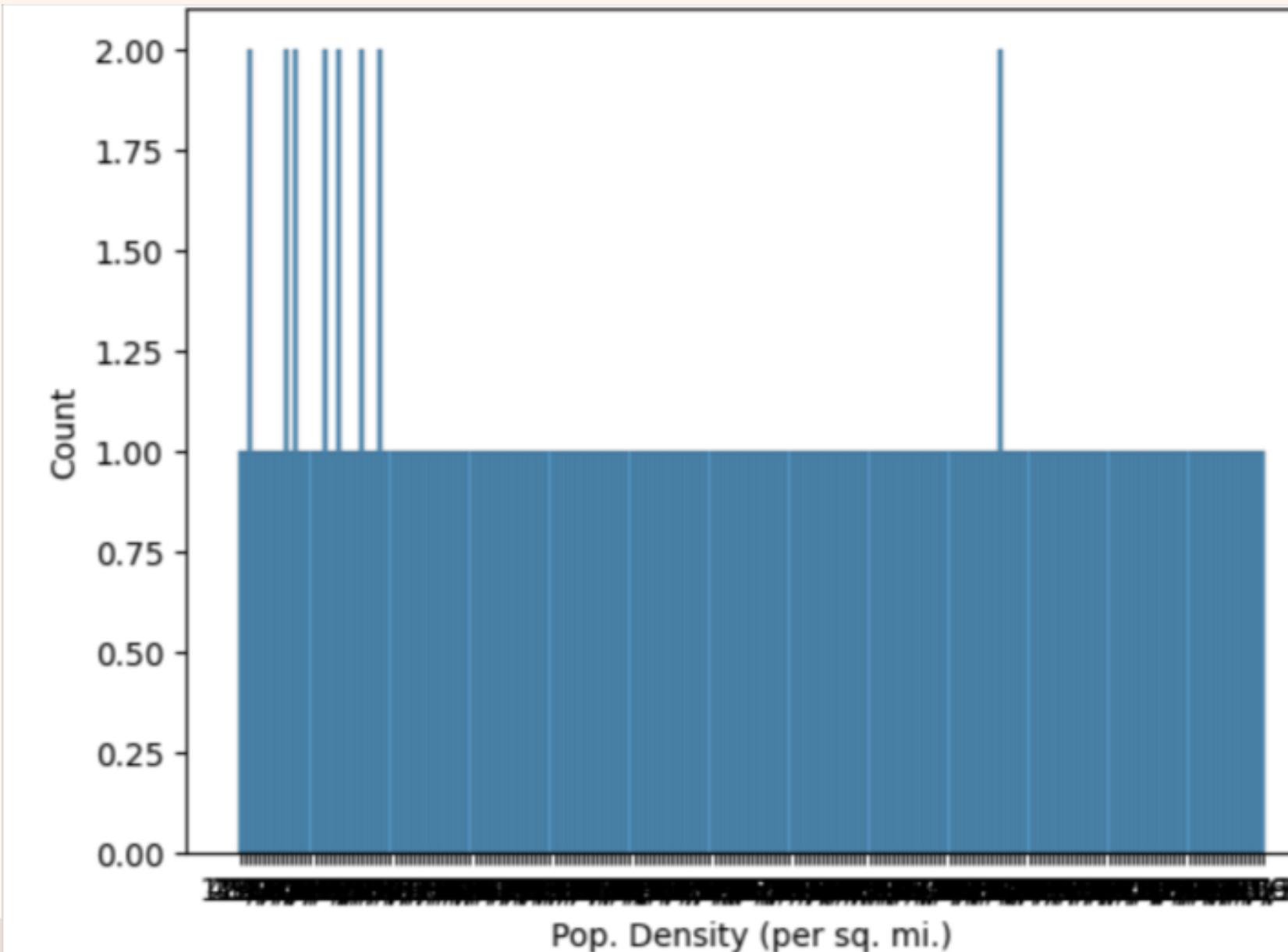
	Country	Region	Population	\
9	Armenia	C.W. OF IND. STATES	2976372	
143	Namibia	SUB-SAHARAN AFRICA	2044147	
15	Bahrain	NEAR EAST	698585	
124	Madagascar	SUB-SAHARAN AFRICA	18595469	
153	N. Mariana Islands	OCEANIA	82459	
...	...		...	...
136	Micronesia, Fed. St.	OCEANIA	108004	
29	Brunei	ASIA (EX. NEAR EAST)	379444	
109	Korea, North	ASIA (EX. NEAR EAST)	23113019	
67	Fiji	OCEANIA	905949	
215	Uruguay	LATIN AMER. & CARIB	3431932	
	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	\
9	29800	99,9	0,00	
143	825418	2,5	0,19	
15	665	1050,5	24,21	
124	587040	31,7	0,82	
153	477	172,9	310,69	
...	...	...	...	...
136	702	153,9	870,66	
29	5770	65,8	2,79	
109	120540	191,8	2,07	
67	18270	49,6	6,18	
215	176220	19,5	0,37	
	Net migration	Infant mortality (per 1000 births)	GDP (\$ per capita)	\
9	-6,47	23,28	3500.0	



# Explorating Data

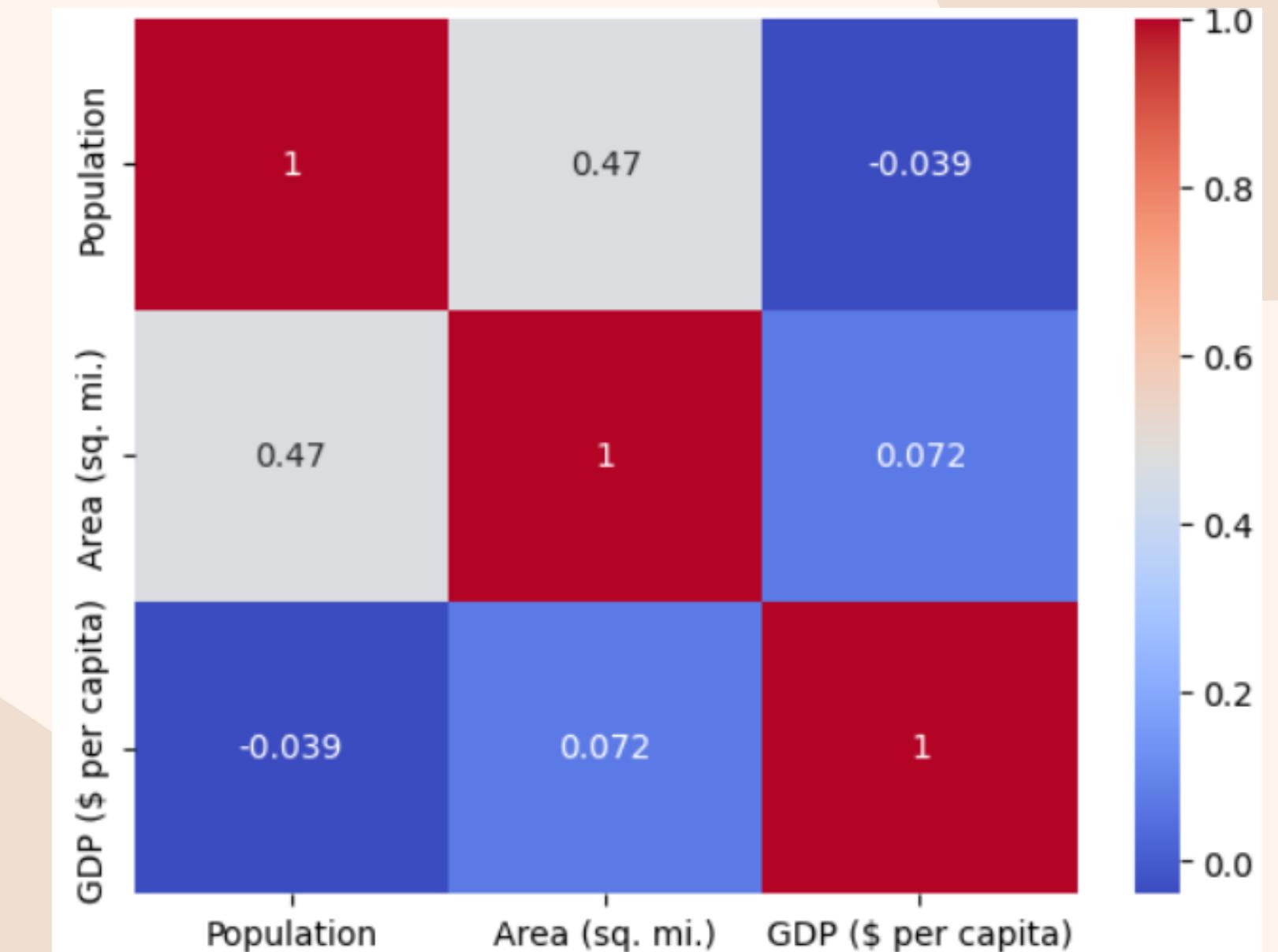


# Visualisasi Comparison

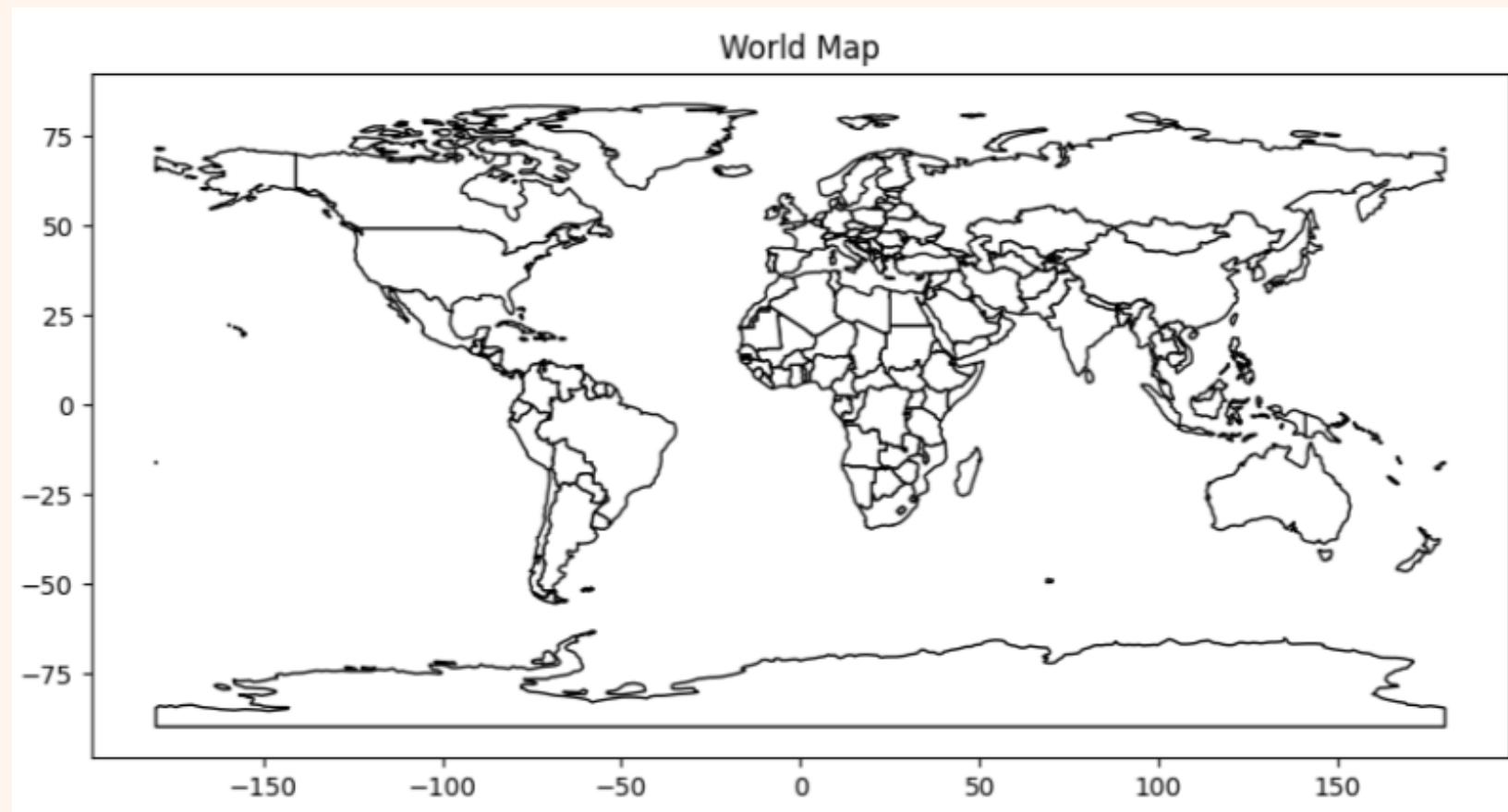


Menampilkan distribusi data Pop density ( Kepadatan penduduk dalam mil persegi). Menggunakan histogram

Korelasi antara variabel popdensity dengan variabel lain

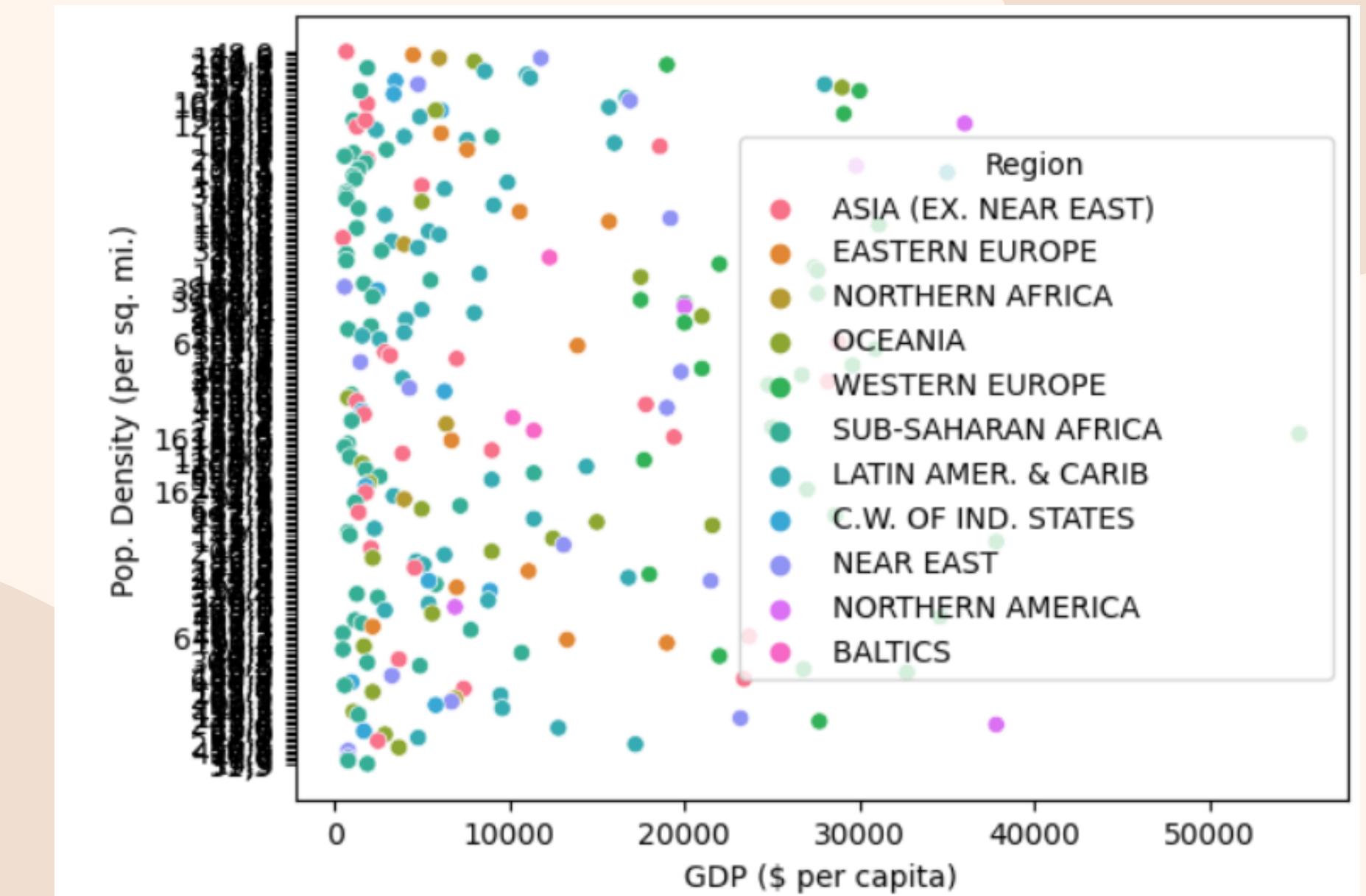


# Visualisasi Comparison

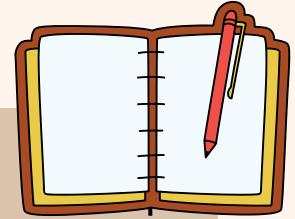


Visualisasi peta dunia dengan menggunakan library Shapefile dan Matplotlib

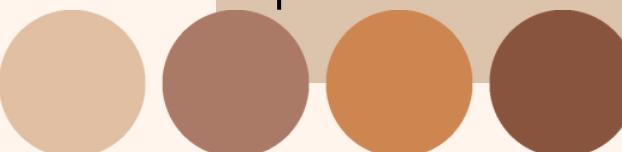
Analisis faktor yang-faktor yang dapat mempengaruhi pop Density



# Data Preparation



**Tahap ini untuk mempersiapkan data untuk analisis, termasuk membersihkan data yang tidak relevan, memperbaiki data yang rusak, dan membuat dataset siap digunakan.**



# Cleansing Data

```
[ ] # Cek missing values  
print(df_sample.isnull().sum())
```

```
Country          0  
Region          0  
Population      0  
Area (sq. mi.)  0  
Pop. Density (per sq. mi.) 0  
Coastline (coast/area ratio) 0  
Net migration    0  
Infant mortality (per 1000 births) 0  
GDP ($ per capita) 0  
Literacy (%)     5  
Phones (per 1000) 0  
Arable (%)       0  
Crops (%)        0  
Other (%)        0  
Climate          6  
Birthrate        0  
Deathrate        0  
Agriculture      1  
Industry          1  
Service           1  
dtype: int64
```

```
[ ] #Menghapus Data Kosong / preprocessing data  
df1 = df_sample.dropna()  
df1
```

## Menghapus data kosong

```
7] 1 #Menghapus Data Kosong / prepocessing data  
2 df1 = df_sample.dropna()  
3 df1
```

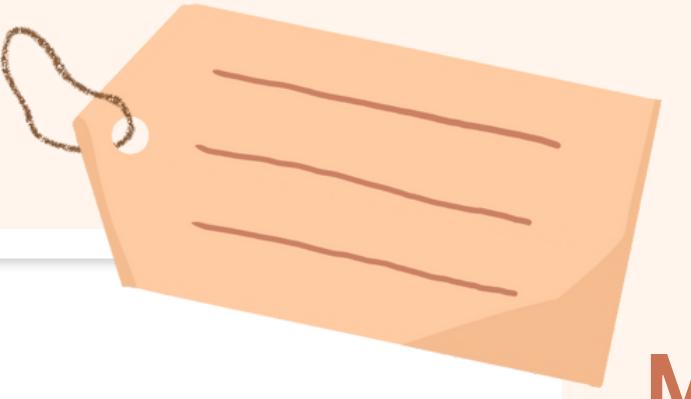
```
9] 1 df1.describe()
```

	Population	Area (sq. mi.)	GDP (\$ per capita)
count	5.700000e+01	5.700000e+01	57.000000
mean	2.316237e+07	3.739939e+05	9394.736842
std	4.265625e+07	5.572873e+05	10086.862408
min	3.398700e+04	1.600000e+02	500.000000
25%	1.136334e+06	1.782000e+04	1700.000000
50%	8.989046e+06	1.184800e+05	5400.000000
75%	2.311302e+07	5.279700e+05	16900.000000
max	2.454527e+08	2.505810e+06	35000.000000

menghitung jumlah nilai null (kosong) dalam setiap kolom dan menghapus semua baris yang memiliki nilai null (missing value)

```
10] 1 #Menghapus kolom yang tidak relevan  
2 df1 = df_sample.drop(['Arable (%)','Phones (per 1000)'], axis=1)  
3 df1
```

# Cleansing Data



## Encoding

Perintah ini akan mengubah nilai pada kolom 'Country' dari string menjadi bilangan numerik yang merepresentasikan tiap nilai unik pada kolom tersebut.

Tujuan dari encoding pada kolom 'Country' ini adalah untuk mempersiapkan data df1 agar dapat diproses oleh algoritma machine learning yang hanya menerima data numerik sebagai input. Setelah encoding dilakukan, kolom 'Country' pada df1 akan berisi nilai numerik yang merepresentasikan tiap negara yang ada pada dataset.

```
1 #Import library
2 from sklearn.preprocessing import LabelEncoder
3
4 # Membuat objek LabelEncoder
5 le = LabelEncoder()
6
7 # Melakukan encoding pada kolom 'Country'
8 df1['Country'] = le.fit_transform(df1['Country'])
9 df1
```

## ENCODING

# MENGHILANGKAN OUTLIERS

## Menghilangkan outliers

Hal ini dilakukan untuk memastikan bahwa data yang digunakan dalam analisis atau model machine learning adalah data yang lebih representatif dan tidak terpengaruh oleh nilai-nilai ekstrem. Dengan menghilangkan outliers, dapat membantu meminimalkan kesalahan analisis dan meningkatkan kinerja model machine learning yang dibangun.

```
1 # Menghitung nilai IQR pada kolom 'GDP ($ per capita)'
2 Q1 = df1['GDP ($ per capita)'].quantile(0.25)
3 Q3 = df1['GDP ($ per capita)'].quantile(0.75)
4 IQR = Q3 - Q1
5
6 # Menghitung nilai batas bawah dan batas atas
7 lower_bound = Q1 - 1.5*IQR
8 upper_bound = Q3 + 1.5*IQR
9
10 # Menghapus baris dengan nilai di luar batas bawah dan batas atas
11 df1 = df1[(df1['GDP ($ per capita)'] >= lower_bound) & (df1['GDP ($ per capita)'] <= upper_bound)]
12 df1
```

```
1 # Menghapus nilai yang duplikat pada kolom 'Country'
2 df1 = df_sample.drop_duplicates(subset=['Country'])
3 df1
```

Scaling adalah suatu proses untuk mengubah skala nilai pada suatu fitur agar nilainya berada pada rentang tertentu. Hal ini dilakukan untuk memastikan bahwa semua fitur memiliki pengaruh yang setara pada proses pembelajaran mesin, sehingga mencegah fitur yang memiliki nilai besar dominan terhadap fitur yang memiliki nilai kecil.

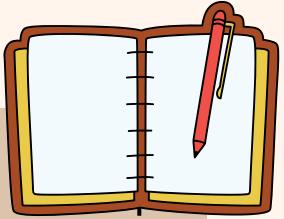
```
1 # Import library
2 from sklearn.preprocessing import MinMaxScaler
3
4
5
6 # Membuat objek MinMaxScaler
7 scaler = MinMaxScaler()
8
9 # Melakukan scaling pada kolom 'Population'
10 df1['Population'] = scaler.fit_transform(df1[['Population']])
11 df1
```

## SCALING

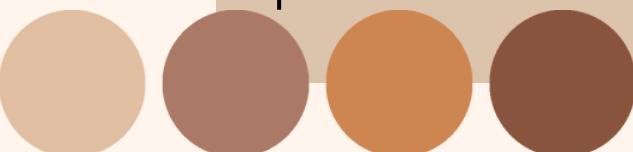
## Menyimpan hasil cleaning ke dataset baru

```
✓ [26] 1 # Simpan cleaned data ke file csv
2 df1.to_csv('clean_countries.csv', index=False)
```

# Modelling



## K-MEANS



```
[27] 1 import pandas as pd
2 import matplotlib.pyplot as plt
3 from sklearn.cluster import KMeans
4
5 def kmeans_clustering(data, n_clusters):
6     # Preprocessing data
7     data = data[['Country', 'Population', 'Area (sq. mi.)']].dropna()
8     data['Density'] = data['Population'] / data['Area (sq. mi.)']
9     X = data[['Density']]
10
11    # Create k-means model
12    kmeans = KMeans(n_clusters=n_clusters, n_init=10, random_state=0)
13
14    # Fit the model to the data
15    kmeans.fit(X)
16
17    # Assign the cluster labels to the original dataset
18    data['Cluster'] = kmeans.labels_
19
20    # Print the countries in each cluster
21    for i in range(n_clusters):
22        print(f'Cluster {i}:')
23        print(data[data['Cluster'] == i]['Country'].values)
24        print()
25
26    # Visualize the clustering result
27    plt.scatter(data['Density'], [0] * len(data), c=data['Cluster'], cmap='viridis')
28    plt.yticks([])
29    plt.xlabel('Population Density (per sq. mi.)')
30    plt.show()
31
32 # Load the dataset
33 data = pd.read_csv('clean_countries.csv', decimal=',')
34
35 # Perform clustering and visualization
36 kmeans_clustering(data, n_clusters=3)
```

```
Cluster 0:  
['Armenia ' 'Namibia ' 'Madagascar ' 'Georgia ' 'Malaysia ' 'Indonesia '  
'Congo, Dem. Rep. ' 'Samoa ' 'Thailand ' 'Belgium ' 'Bulgaria '  
'Seychelles ' 'Liberia ' 'Guinea ' 'Guatemala ' 'Sudan ' 'Liechtenstein '  
'Zambia ' 'Peru ' 'Belarus ' 'Djibouti ' 'Iraq ' 'Reunion ' 'Swaziland '  
'Saint Lucia ' 'Gambia, The ' 'Greece ' 'Kyrgyzstan ' 'Panama '  
'Nigeria ' 'Netherlands ' 'Malawi ' 'Guadeloupe ' 'Mozambique ' 'France '  
'Kuwait ' 'Iceland ' 'Egypt ' 'Portugal ' 'Finland ' 'Sierra Leone '  
'Uzbekistan ' 'Yemen ' 'Turkey ' 'Ukraine ' 'Cayman Islands ' 'Bolivia '  
'Sweden ' 'Aruba ' 'Micronesia, Fed. St. ' 'Brunei ' 'Korea, North '  
'Fiji ' 'Uruguay ']
```

Cluster 1:

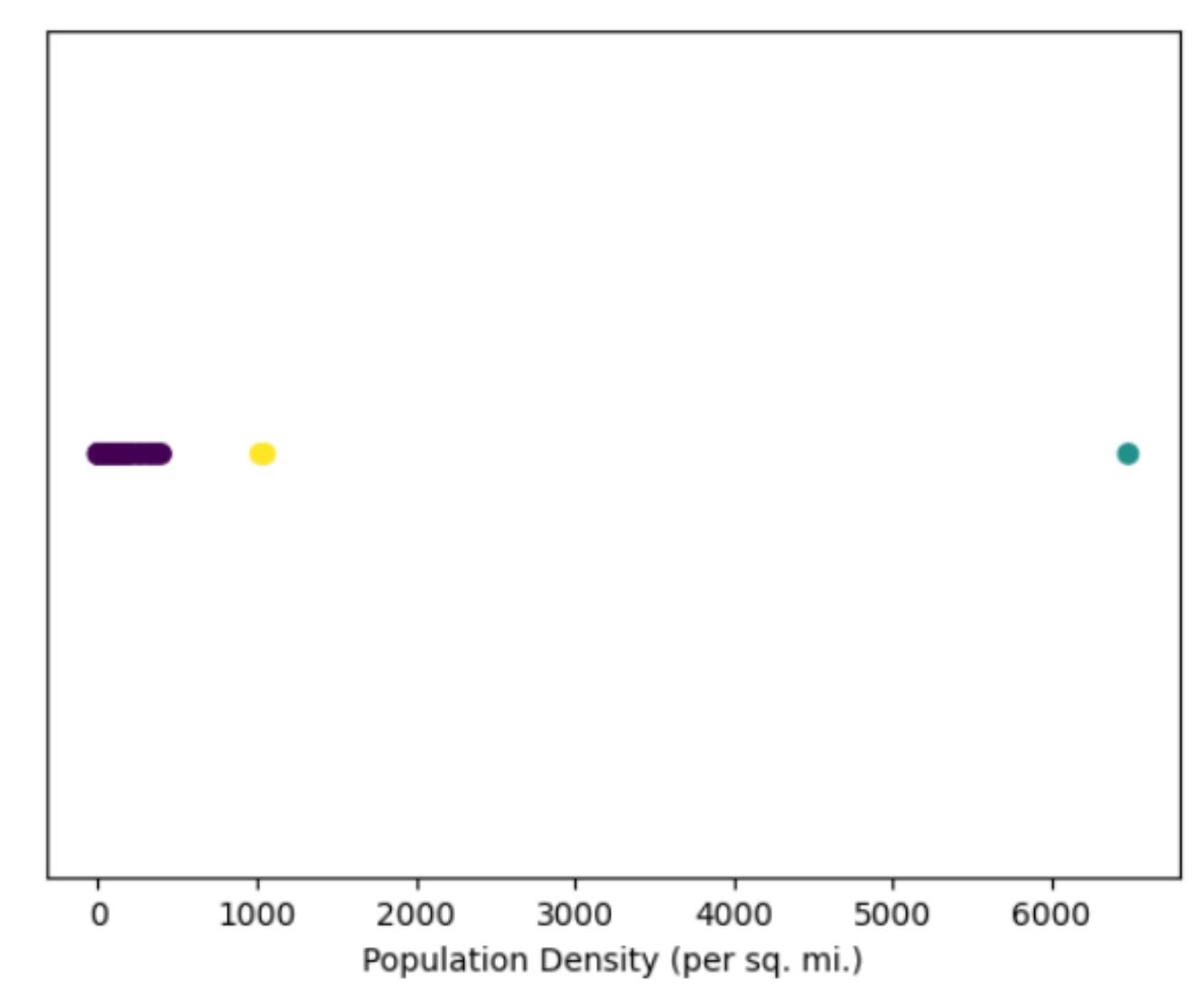
```
['Singapore ']
```

Cluster 2:

```
['Bahrain ' 'Bangladesh '']
```

negara-negara telah berhasil dikelompokkan menjadi tiga kelompok berdasarkan tingkat kepadatan penduduknya.

- Cluster 0 berisi negara-negara dengan kepadatan penduduk paling tinggi
- cluster 1 berisi negara-negara dengan kepadatan penduduk sedang
- cluster 2 berisi negara-negara dengan kepadatan penduduk paling rendah

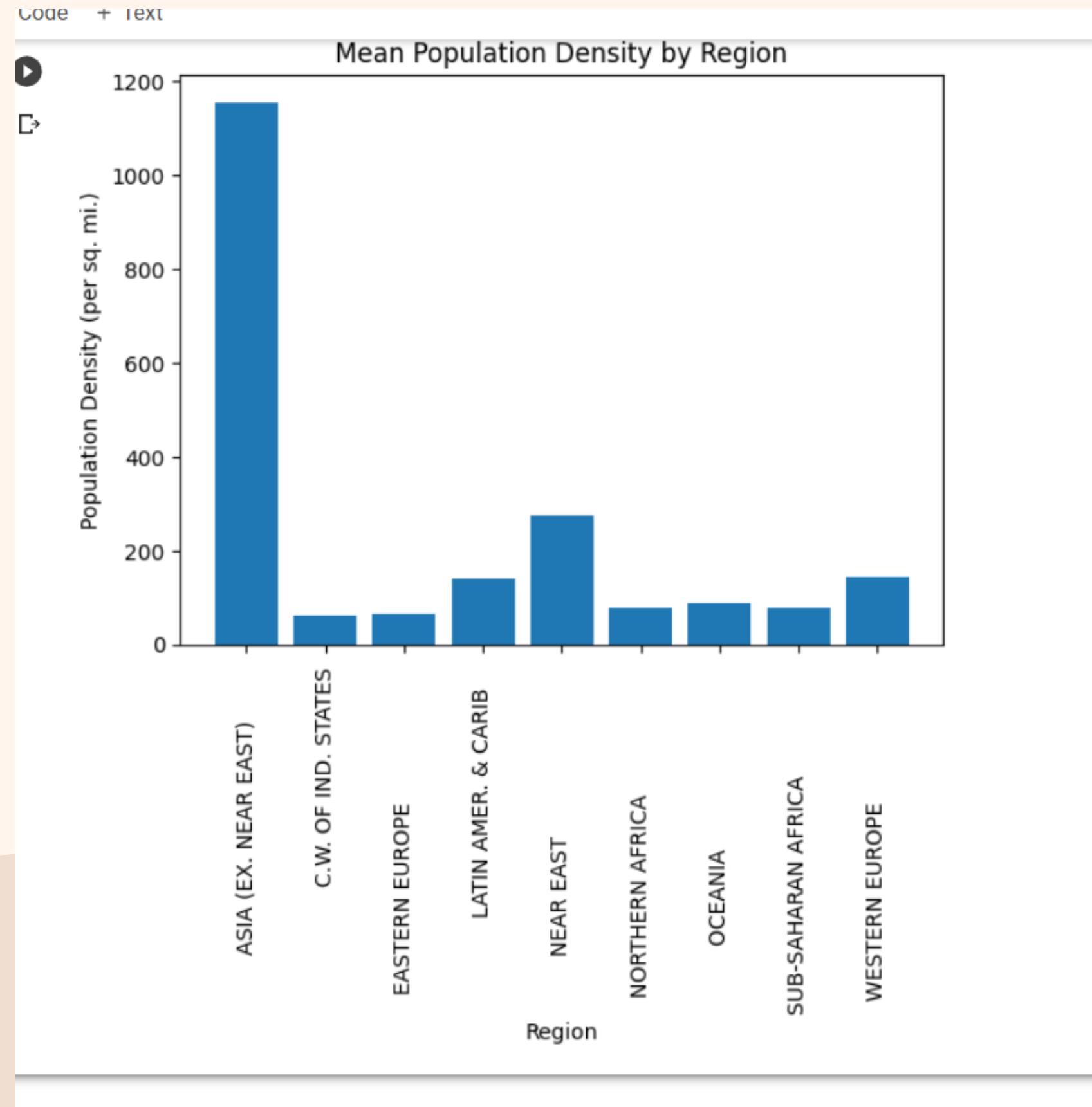


# RATA RATA KEPADATAN PENDUDUK

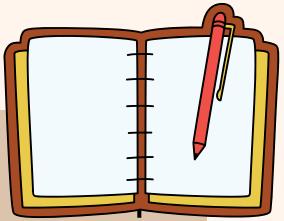
```
1 import pandas as pd
2
3 # Load dataset
4 data = pd.read_csv('clean_countries.csv', decimal=',')
5
6 # Drop rows with null values
7 data = data.dropna()
8
9 # Calculate population density
10 data['Density'] = data['Population'] / data['Area (sq. mi.)']
11
12 # Group data by region and calculate mean density
13 mean_density = data.groupby('Region')['Density'].mean()
14
15 # Round the mean density values to 2 decimal places
16 mean_density = round(mean_density, 2)
17
18 # Display the mean density table
19 print(mean_density)
```

Region	
ASIA (EX. NEAR EAST)	1155.80
C.W. OF IND. STATES	63.50
EASTERN EUROPE	66.59
LATIN AMER. & CARIB	141.90
NEAR EAST	275.67
NORTHERN AFRICA	78.77
OCEANIA	87.84
SUB-SAHARAN AFRICA	77.76
WESTERN EUROPE	143.89
Name: Density, dtype: float64	

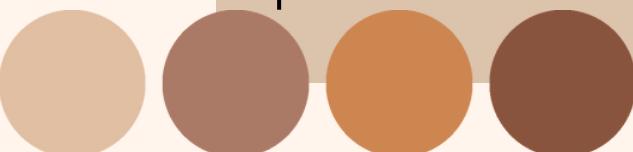
# Hasil Visual



# EVALUATION



## K-MEANS

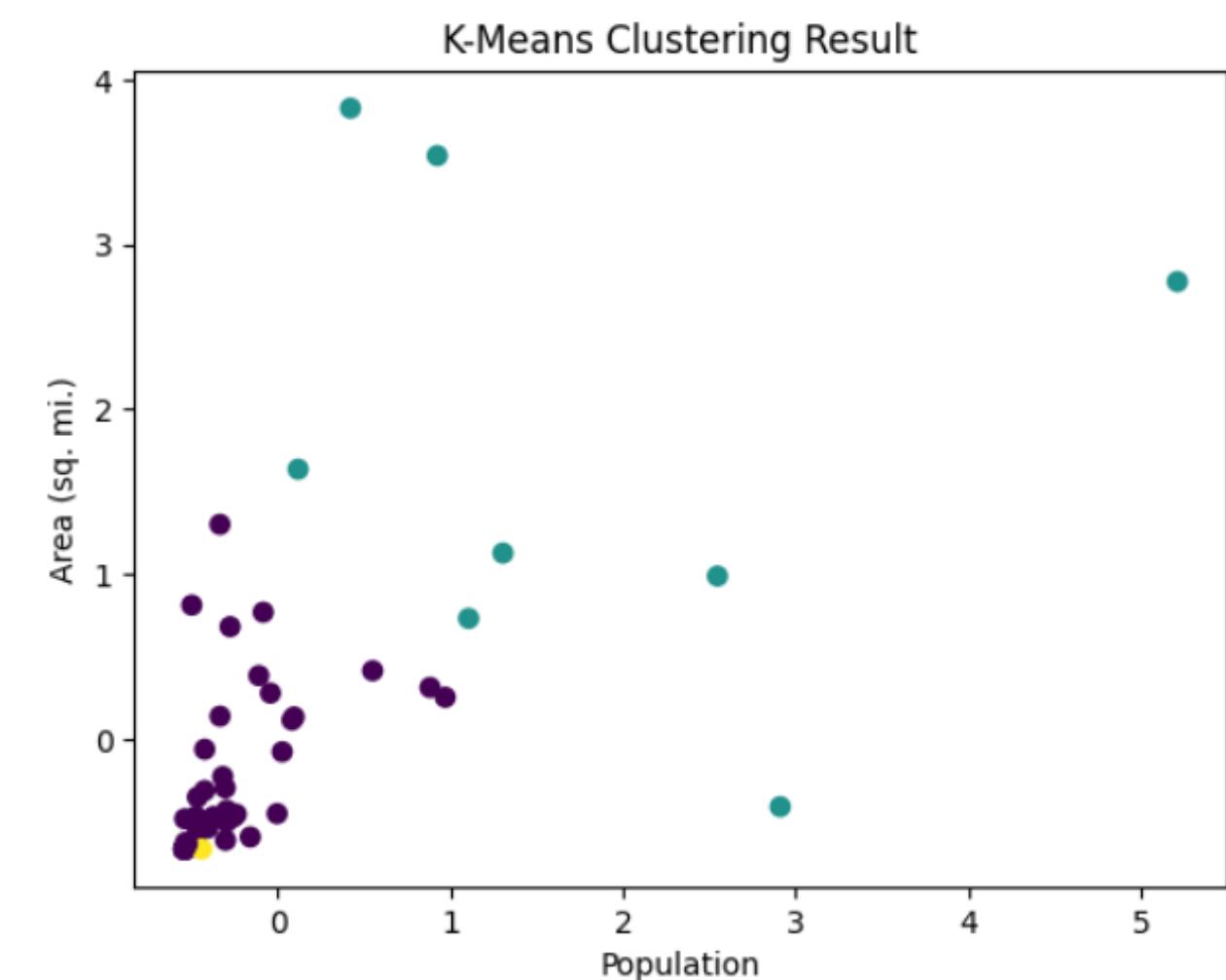




## ▼ Evaluation

```
✓ [49] 1 import pandas as pd
2
3 # Load the dataset
4 data = pd.read_csv('clean_countries.csv', decimal=',')  
  
✓ [50] 1 # Preprocessing data
2 data = data[['Country', 'Population', 'Area (sq. mi.)']].dropna()
3 data['Density'] = data['Population'] / data['Area (sq. mi.)']
4 X = data[['Density']]  
  
✓ [51] 1 data = data.dropna() #menghapus baris atau data yang memiliki nilai null
2 X = data.drop(['Country'], axis=1) # Drop the 'Country' column
3 X = (X - X.mean()) / X.std() # melakukan normalisasi atau scaling pada data numerik pada dataset  
  
✓ [52] 1 from sklearn.cluster import KMeans
2
3 k = 3 # menentukan jumlah klaster yaitu 3
4 kmeans = KMeans(n_clusters=k, random_state=0)
5 kmeans.fit(X) # melakukan fitting atau training model KMeans dengan dataset  
  
KMeans
KMeans(n_clusters=3, random_state=0)
```

```
1 import matplotlib.pyplot as plt
2
3 plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=kmeans.labels_, cmap='viridis')
4 plt.xlabel(X.columns[0])
5 plt.ylabel(X.columns[1])
6 plt.title('K-Means Clustering Result')
7 plt.show()
```



# Hasil Akhir

```
1 import pandas as pd
2 from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score
3
4 # Calculate silhouette score
5 silhouette = silhouette_score(X, kmeans.labels_)
6
7 # Calculate Calinski-Harabasz index
8 calinski = calinski_harabasz_score(X, kmeans.labels_)
9
10 # Calculate Davies-Bouldin index
11 davies_bouldin = davies_bouldin_score(X, kmeans.labels_)
12
13 # Create a DataFrame to store the scores
14 scores = pd.DataFrame({'Silhouette Score': [silhouette],
15                         'Calinski-Harabasz Score': [calinski],
16                         'Davies-Bouldin Score': [davies_bouldin]})
17
18 # Display the scores
19 print(scores)
20
```

	Silhouette Score	Calinski-Harabasz Score	Davies-Bouldin Score
0	0.662522	56.102368	0.636983

1. Nilai ini merepresentasikan seberapa dekat suatu titik data dengan klaster yang sama dibandingkan dengan klaster lainnya. Nilai Silhouette Score yang mendekati 1 menunjukkan bahwa pengelompokan atau clustering yang dilakukan sangat baik.

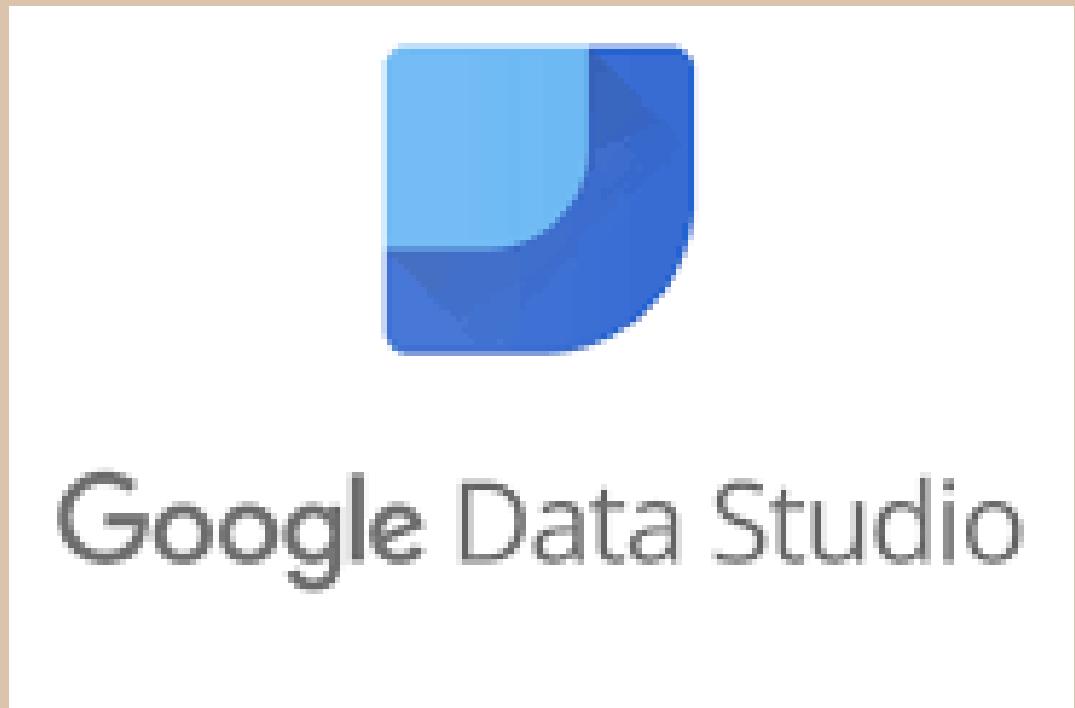
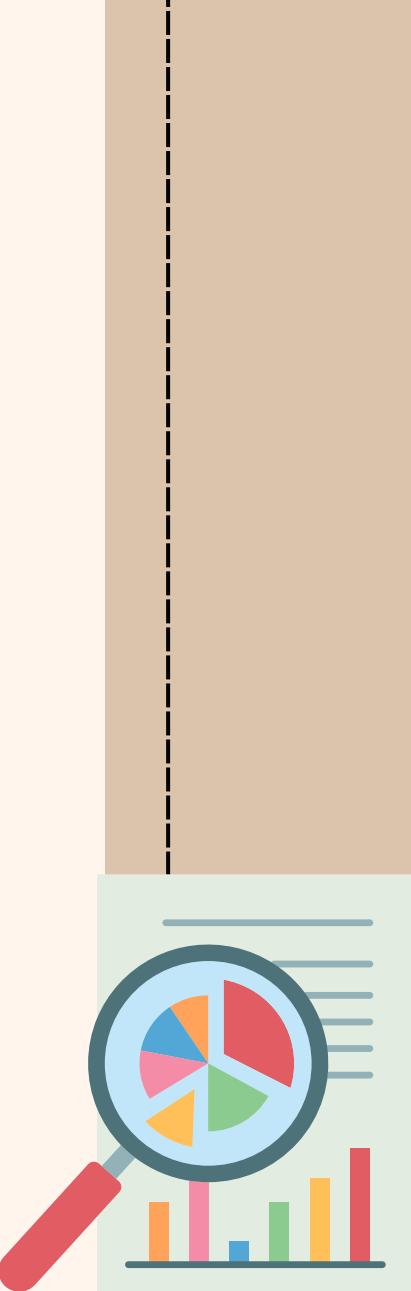
2. Dalam kasus ini, nilai Calinski-Harabasz score sebesar 56.102 Nilai ini merepresentasikan seberapa baik atau seberapa terpisah klaster yang dihasilkan oleh model KMeans.

3. Semakin rendah nilai Davies-Bouldin score, semakin baik kualitas cluster yang dihasilkan. Dalam kasus ini, nilai Davies-Bouldin score sebesar 0.636 menunjukkan bahwa kualitas cluster yang dihasilkan cukup baik

## KESIMPULAN

Berdasarkan ketiga nilai tersebut, dapat disimpulkan bahwa k-means yang dilakukan pada dataset Countries of the World cukup efektif dalam membentuk cluster dengan  $k=3$  karena dengan nilai Silhouette Score yang mendekati 1, nilai Calinski-Harabasz Score yang cukup tinggi, dan nilai Davies-Bouldin Score yang cukup rendah.

# Deployment



# UNSUPERVISED

**Countries of the World**

Kumpulan data yang berisi informasi mengenai berbagai negara di seluruh dunia. Bertujuan untuk mempelajari dan memahami perbedaan antara negara-negara di seluruh dunia dan faktor-faktor yang memengaruhi kesejahteraan penduduk di negara-negara tersebut.

**VISUALISASI NEGARA BERDASARKAN KEPADATAN PENDUDUK**

Google Pintasan keyboard Data peta ©2023 Syarat Penggunaan

**VISUALISASI** Persentase populasi yang bisa membaca dan menulis berdasarkan region

**GRAFIK** Jumlah kelahiran dan kematian per 1000 penduduk dalam setahun

**VISUALISASI** Luas Daerah per negara

**Visualisasi** sebaran data pada klaster

**Tabel Persentase PDB yang berasal dari sektor pertanian,Industri,Jasa**

GDP (\$ pe...	Agricultur...	Ser...	Indus...
1. 1000	769	177	54
2... 1700	650	592	371
3... 1900	586	644	401
4... 800	433	1441	666
5... 1600	353	439	208
6... 600	342	499	158
7... 2000	289	559	152
8... 900	269	244	487
9... 1200	262	39	348
1... 3500	239	418	343
1... 2100	237	401	362
1... 4100	227	585	188
1... 5400	194	434	454
1... 1300	182	632	259
1... 2500	172	553	275
1... 4000	149	493	357
1... 3200	134	408	458

Activate Win  
Go to Settings to

# Table Kontribusi

TAHAPAN	KETERANGAN	SUPERVISED	UNSUPERVISED
Business Understanding	Penjelasan	Reyhan	Windy
	Penjelasan kolom	Novi	Windy
	Tujuan	Novi	Windy
Data Understanding	Explorating Data	Novi	Novi
	Visualisasi Comparison	Novi	Novi
	Visualisasi Distribution	Novi	Novi
Data Preparation	Cleansing Data	Windy	Novi
	Membagi Data Uji	Windy	Novi
Modelling	Naive Bayes	Novi	
	KNN	Reyhan	
	C4.5	Windy	
	K-MEANS		Novi
Evaluation		Windy	Novi
Deployment	Google Data Studio	Novi	
	PPT	Windy & Reyhan	

# THANK YOU