UJIAN AKHIR SEMESTER

PREDIKSI KELAS HARGA PONSEL BERDASARKAN SPESIFIKASI

Dosen Pengampu:
Theopilus Bayu Sasongko, S.Kom, M.Eng

Disusun Oleh:

Rachmasari Annisa Rida	22.11.4624
Suci Prasetia Ningrum	22.11.5116
Nur Ahmad Fathi BR	22.11.5267
Muhammad Armin	22.11.5281 (tidak kontribusi)
Nurrochim Amin Putra	22.11.5299

PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS AMIKOM YOGYAKARTA

2025

Latar Belakang

Perkembangan teknologi ponsel pintar (smartphone) semakin pesat dalam beberapa dekade terakhir. Dengan hadirnya berbagai merek dan model ponsel yang menawarkan spesifikasi unik di rentang harga yang berbeda, konsumen seringkali merasa kesulitan untuk menentukan pilihan yang paling sesuai dengan kebutuhan dan anggaran mereka. Di sisi lain, produsen ponsel menghadapi tantangan untuk memahami preferensi pasar dan merancang produk yang dapat menarik minat konsumen di segmen tertentu.

Tingginya persaingan di industri teknologi ponsel juga menuntut produsen untuk mampu mengidentifikasi spesifikasi apa yang menjadi prioritas konsumen. Selain itu, konsumen tidak selalu memiliki informasi yang memadai untuk menilai apakah harga yang ditawarkan sesuai dengan nilai spesifikasi ponsel tersebut. Akibatnya, konsumen sering kali bergantung pada rekomendasi pihak ketiga atau ulasan yang belum tentu berbasis data.

Dalam dunia yang semakin data-driven, pemanfaatan analisis data dan pembelajaran mesin (machine learning) memberikan peluang besar untuk memahami hubungan antara spesifikasi teknis sebuah ponsel dan kategori harganya. Analisis ini tidak hanya membantu konsumen dalam membuat keputusan yang lebih baik, tetapi juga memberikan wawasan berharga bagi produsen dalam menyusun strategi pengembangan produk dan pemasaran yang lebih efektif.

Dari analisis yang dilakukan, diharapkan akan memudahkan konsumen dalam pengambilan keputusan terkait pemilihan *smartphone* yang sesuai dengan spesifikasi, kebutuhan, dan anggaran yang konsumen miliki.

Metode Penelitian

Metode penelitian yang kami digunakan dalam penelitian ini adalah metode pendekatan kuantitatif, yang bertujuan untuk menganalisis hubungan antara variabel spesifikasi ponsel (seperti RAM, daya baterai, kamera, dan lainnya) dengan variabel target berupa kategori harga (Price Range). Tahapan penelitian kami dimulai dengan pengambilan dataset train.csv melalui platform kaggle selanjutnya kami melakukan pengolahan data tersebut hingga sampai evaluasi model.

Dataset

Penelitian kami ini menggunakan dataset yang kami ambil dari situs Kaggle. Dataset ini terdiri atas 2000 data dan 21 kolom yaitu battery power, blue, clock_speed, dual_sim, fc, four_g, int_memory, m_dep, mobile_wt, n_cores, pc, px_height, px_width, ram, sc_h, sc_w, talk_time, three_g, touch_screen, wifi, price_range. Dataset tersebut terakhir diperbarui sekitar setahun yang lalu. Untuk penjelasan masing-masing kolom dapat dilihat pada link dataset berikut :https://www.kaggle.com/datasets/jacksondivakarr/phone-classification-dataset

Pre-Processing

```
print("Tipe Data Setiap Kolom:")
print(data.dtypes)
```

```
Tipe Data Setiap Kolom:
battery_power      int64
blue               int64
clock_speed      float64
dual_sim           int64
fc                 int64
four_g             int64
int_memory         int64
m_dep            float64
mobile_wt          int64
n_cores            int64
pc                 int64
px_height          int64
px_width           int64
ram                int64
sc_h               int64
sc_w               int64
talk_time          int64
three_g            int64
touch_screen       int64
wifi               int64
price_range        int64
dtype: object
```

Langkah pertama yang dilakukan dalam pre-processing adalah menampilkan tipe data pada setiap kolom dengan tujuan untuk mengetahui tipe data apa saja yang ada dari kolom-kolom pada DataFrame. Dapat terlihat bahwa dari 21 kolom, 19 diantaranya berupa kolom dengan tipe data integer, sedangkan 2 kolom lainnya yaitu “clock_speed” dan “m_dep” berupa kolom dengan tipe data float.

```
[ ]  print("\nNama Kolom Sebelum Diganti:")
     print(pd.DataFrame({'Kolom Asli' : data.columns}))
```

```
Nama Kolom Sebelum Diganti:
       Kolom Asli
0   battery_power
1            blue
2     clock_speed
3        dual_sim
4              fc
5          four_g
6      int_memory
7           m_dep
8       mobile_wt
9         n_cores
10             pc
11      px_height
12       px_width
13            ram
14           sc_h
15           sc_w
16      talk_time
17        three_g
18   touch_screen
19           wifi
20    price_range
```

Langkah selanjutnya, menampilkan seluruh nama kolom yang ada dengan mengubah nama DataFrame data (DataFrame lama) menjadi “Kolom Asli”, sebelum nantinya nama kolom akan diubah.

```
data.rename(columns={
    'battery_power': 'Battery_Power',
    'blue': 'Bluetooth',
    'clock_speed': 'Clock_Speed',
    'dual_sim': 'Dual_SIM',
    'fc': 'Front_Camera',
    'four_g': '4G',
    'int_memory': 'Internal_Memory',
    'm_dep': 'Mobile_Depth',
    'mobile_wt': 'Mobile_Weight',
    'n_cores': 'Num_Cores',
    'pc': 'Primary_Camera',
    'px_height': 'Pixel_Height',
    'px_width': 'Pixel_Width',
    'ram': 'RAM',
    'sc_h': 'Screen_Height',
    'sc_w': 'Screen_Width',
    'talk_time': 'Talk_Time',
    'three_g': '3G',
    'touch_screen': 'Touch_Screen',
    'wifi': 'WiFi',
    'price_range': 'Price_Range'
}, inplace=True)\

print("\nNama Kolom Setelah Diganti:")
print(pd.DataFrame({'Kolom Baru': data.columns}))
```

```
Nama Kolom Setelah Diganti:
         Kolom Baru
0     Battery_Power
1         Bluetooth
2       Clock_Speed
3          Dual_SIM
4      Front_Camera
5                4G
6   Internal_Memory
7      Mobile_Depth
8     Mobile_Weight
9         Num_Cores
10   Primary_Camera
11     Pixel_Height
12      Pixel_Width
13              RAM
14    Screen_Height
15     Screen_Width
16        Talk_Time
17               3G
18     Touch_Screen
19             WiFi
20      Price_Range
```

Setelah menampilkan DataFrame dengan data nama kolom yang lama, selanjutnya mengganti nama kolom-kolom pada DataFrame data berdasarkan dictionary yang diberikan dengan format (‘nama kolom lama’ : ‘nama kolom baru’) , lalu menampilkan nama-nama kolom yang baru setelah dilakukan penggantian nama kolom.

```
[ ]  print("\nJumlah Nilai Null Per Kolom:")
     print(data.isnull().sum())
```

```
Jumlah Nilai Null Per Kolom:
Battery_Power      0
Bluetooth          0
Clock_Speed        0
Dual_SIM           0
Front_Camera       0
4G                 0
Internal_Memory    0
Mobile_Depth       0
Mobile_Weight      0
Num_Cores          0
Primary_Camera     0
Pixel_Height       0
Pixel_Width        0
RAM                0
Screen_Height      0
Screen_Width       0
Talk_Time          0
3G                 0
Touch_Screen       0
WiFi               0
Price_Range        0
dtype: int64
```

Langkah selanjutnya memeriksa nilai null pada setiap kolom dalam DataFrame data dengan menggunakan perintah fungsi isnull(), kemudian menghitung jumlah nilai null pada setiap kolom dengan menggunakan perintah fungsi sum(), dan terakhir menampilkan hasilnya berupa daftar jumlah nilai null untuk setiap kolom.

```
# Menampilkan jumlah dari setiap label dalam dataset
if 'Price_Range' in data.columns:
    print("\nJumlah Setiap Label di Kolom 'Price_Range':
    print(data['Price_Range'].value_counts())
else:
    print("Kolom 'Price_Range' tidak ditemukan dalam dat
```

```
Jumlah Setiap Label di Kolom 'Price_Range':
Price_Range
1    500
2    500
3    500
0    500
Name: count, dtype: int64
```

Selanjutnya memeriksa kolom 'Price_Range' dalam dataset data dan menampilkan jumlah kemunculan setiap label (kategori) dalam kolom tersebut. Jika kolom 'Price_Range' ada, maka data['Price_Range'].value_counts() akan menghitung dan mencetak jumlah masing-masing kategori atau kelas dalam kolom tersebut. Jika kolom tidak ditemukan, pesan "Kolom 'Price_Range' tidak ditemukan dalam dataset!" akan ditampilkan, memberi tahu bahwa kolom yang dimaksud tidak ada.

```
data['Price_Range'] = data['Price_Range'].astype('catego
print("\nTipe Data Setelah Diubah:")
print(data.dtypes)
```

```
Tipe Data Setelah Diubah:
Battery_Power         int64
Bluetooth             int64
Clock_Speed         float64
Dual_SIM              int64
Front_Camera          int64
4G                    int64
Internal_Memory       int64
Mobile_Depth        float64
Mobile_Weight         int64
Num_Cores             int64
Primary_Camera        int64
Pixel_Height          int64
Pixel_Width           int64
RAM                   int64
Screen_Height         int64
Screen_Width          int64
Talk_Time             int64
3G                    int64
Touch_Screen          int64
WiFi                  int64
Price_Range        category
dtype: object
```

Lalu mengubah tipe data dari kolom ‘Price_Range’ yang sebelumnya berupa integer, diubah menjadi tipe bata category. Dilanjutkan dengan menampilkan seluruh tipe data kolom pada DataFrame data. Setelah dilakukan perubahan, dapat diketahui bahwa terdapat 18 kolom dengan tipe data integer, 2 kolom dengan tipe data float, dan 1 kolom dengan tipe data category yang baru saja dilakukan perubahan.

```
def display_summary(data):
    print("\nSummary Statistik Data Numerik:")
    print(data.describe())

    print("\nSummary Statistik Data Kategorikal:")
    print(data.describe(include=['category']))

display_summary(data)
```

```
Summary Statistik Data Numerik:
       Battery_Power  Bluetooth  Clock_Speed     Dual_SIM  Front_Camera  \
count    2000.000000  2000.0000  2000.000000  2000.000000   2000.000000
mean     1238.518500     0.4950     1.522250     0.509500      4.309500
std       439.418206     0.5001     0.816004     0.500035      4.341444
min       501.000000     0.0000     0.500000     0.000000      0.000000
25%       851.750000     0.0000     0.700000     0.000000      1.000000
50%      1226.000000     0.0000     1.500000     1.000000      3.000000
75%      1615.250000     1.0000     2.200000     1.000000      7.000000
max      1998.000000     1.0000     3.000000     1.000000     19.000000

                4G  Internal_Memory  Mobile_Depth  Mobile_Weight    Num_Cores  \
count  2000.000000      2000.000000   2000.000000    2000.000000  2000.000000
mean      0.521500        32.046500      0.501750     140.249000     4.520500
std       0.499662        18.145715      0.288416      35.399655     2.287837
min       0.000000         2.000000      0.100000      80.000000     1.000000
25%       0.000000        16.000000      0.200000     109.000000     3.000000
50%       1.000000        32.000000      0.500000     141.000000     4.000000
75%       1.000000        48.000000      0.800000     170.000000     7.000000
max       1.000000        64.000000      1.000000     200.000000     8.000000

       Primary_Camera  Pixel_Height  Pixel_Width          RAM  Screen_Height  \
count     2000.000000   2000.000000  2000.000000  2000.000000    2000.000000
mean         9.916500    645.108000  1251.515500  2124.213000      12.306500
std          6.064315    443.780811   432.199447  1084.732044       4.213245
min          0.000000      0.000000   500.000000   256.000000       5.000000
25%          5.000000    282.750000   874.750000  1207.500000       9.000000
50%         10.000000    564.000000  1247.000000  2146.500000      12.000000
75%         15.000000    947.250000  1633.000000  3064.500000      16.000000
max         20.000000   1960.000000  1998.000000  3998.000000      19.000000
```

```
       Screen_Width    Talk_Time           3G  Touch_Screen         WiFi
count   2000.000000  2000.000000  2000.000000   2000.000000  2000.000000
mean       5.767000    11.011000     0.761500      0.503000     0.507000
std        4.356398     5.463955     0.426273      0.500116     0.500076
min        0.000000     2.000000     0.000000      0.000000     0.000000
25%        2.000000     6.000000     1.000000      0.000000     0.000000
50%        5.000000    11.000000     1.000000      1.000000     1.000000
75%        9.000000    16.000000     1.000000      1.000000     1.000000
max       18.000000    20.000000     1.000000      1.000000     1.000000

Summary Statistik Data Kategorikal:
        Price_Range
count          2000
unique            4
top               0
freq            500
```

Selanjutnya membuat fungsi “display_summary” untuk menampilkan ringkasan statistik dari data numerik dan kategorikal dalam DataFrame, dan kemudian memanggil fungsi tersebut dengan DataFrame data sebagai parameternya. Outputnya berupa ringkasan statistik dari data numerik dan kategorikal dalam DataFrame data. Dan untuk hasil pembahasan diatas dapat disimpulkan kalau cont : 2000, unique : 4, top : 0, dan freq : 400.

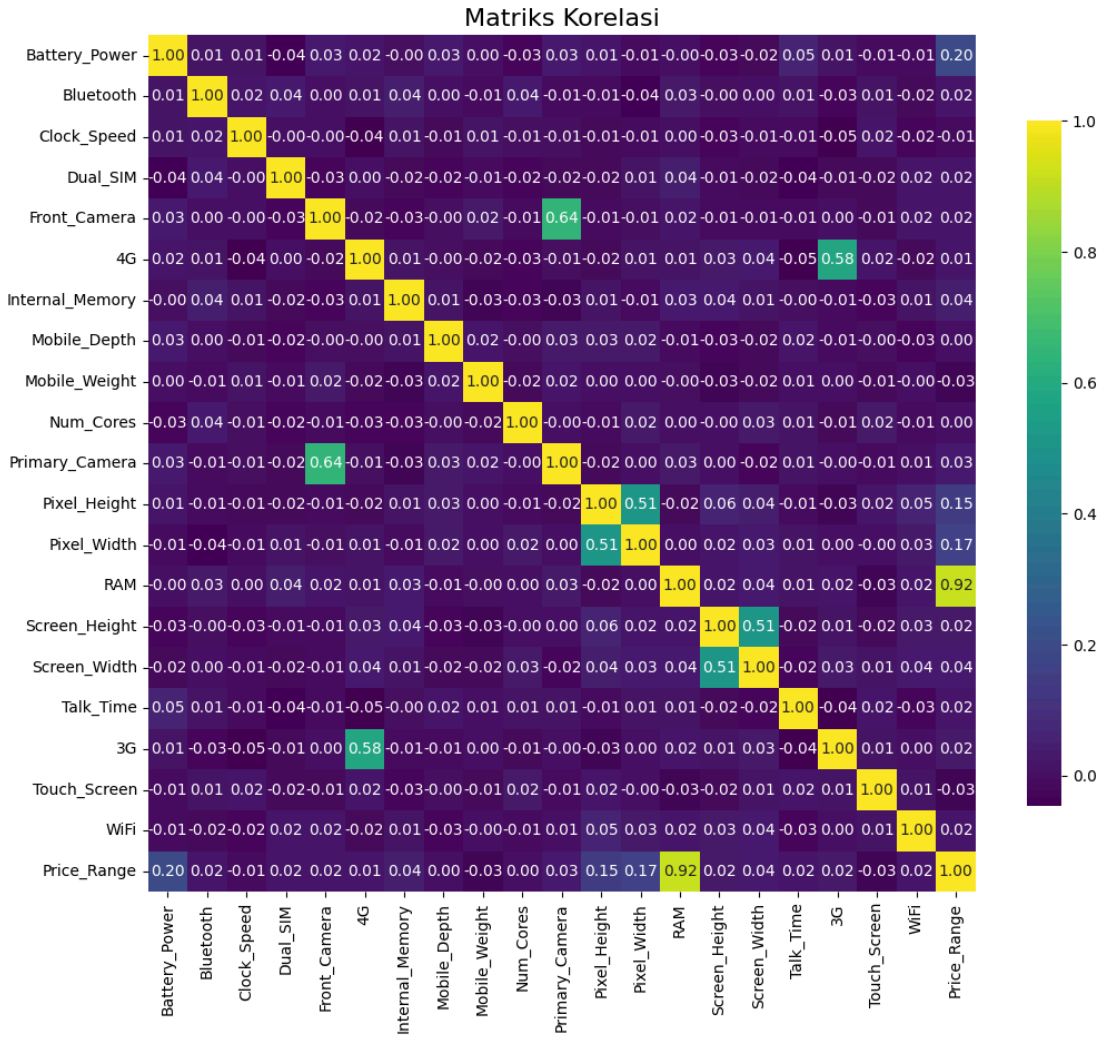
```
[ ]
    # Menampilkan matriks korelasi
    print("\nMatriks Korelasi:")
    correlation_matrix = data.corr()
    print(correlation_matrix)
```

```
                Touch_Screen      WiFi  Price_Range
Battery_Power       -0.010516 -0.008343     0.200723
Bluetooth            0.010061 -0.021863     0.020573
Clock_Speed          0.019756 -0.024471    -0.006606
Dual_SIM            -0.017117  0.022740     0.017444
Front_Camera        -0.014828  0.020085     0.021998
4G                   0.016758 -0.017620     0.014772
Internal_Memory     -0.026999  0.006993     0.044435
Mobile_Depth        -0.002638 -0.028353     0.000853
Mobile_Weight       -0.014368 -0.000409    -0.030302
Num_Cores            0.023774 -0.009964     0.004399
Primary_Camera      -0.008742  0.005389     0.033599
Pixel_Height         0.021891  0.051824     0.148858
Pixel_Width         -0.001628  0.030319     0.165818
RAM                 -0.030455  0.022669     0.917046
Screen_Height       -0.020023  0.025929     0.022986
Screen_Width         0.012720  0.035423     0.038711
Talk_Time            0.017196 -0.029504     0.021859
3G                   0.013917  0.004316     0.023611
Touch_Screen         1.000000  0.011917    -0.030411
WiFi                 0.011917  1.000000     0.018785
Price_Range         -0.030411  0.018785     1.000000

[21 rows x 21 columns]
```

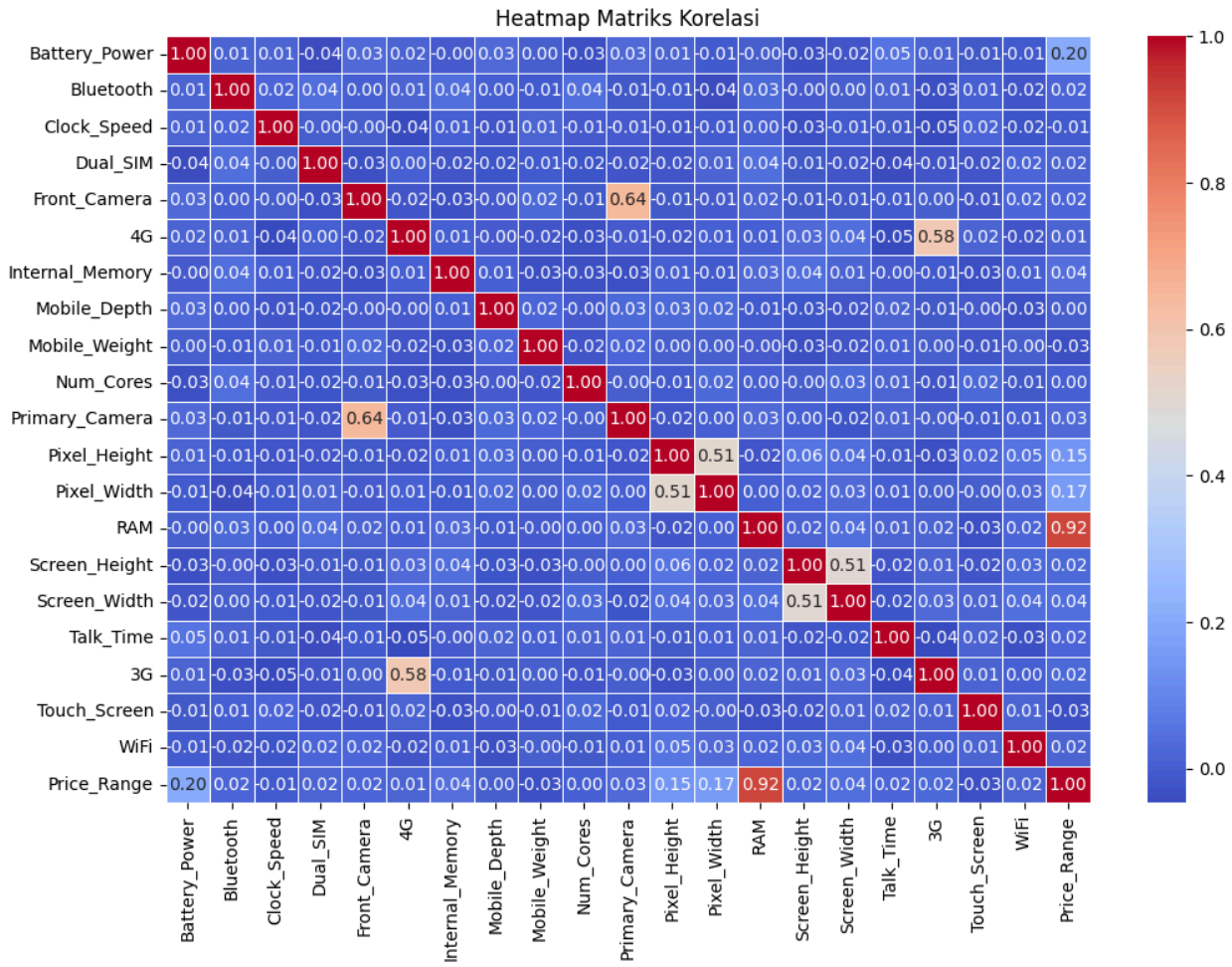
Menghitung dan menampilkan matriks korelasi dari DataFrame data. Matriks korelasi menunjukkan hubungan linear antara setiap pasangan kolom numerik dalam dataset, dengan nilai korelasi antara -1 hingga 1. Nilai mendekati 1 menunjukkan korelasi positif yang kuat, nilai mendekati -1 menunjukkan korelasi negatif yang kuat, dan nilai mendekati 0 menunjukkan tidak adanya korelasi linear. Dan dari hasil diatas dengan 21 row dan 21 kolom, dengan hasil 1 dimiliki oleh 'Touch_Screen, Wifi, Price_Range'.

```
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix,
            annot=True,
            cmap='viridis',
            fmt=".2f",
            cbar_kws={'shrink': 0.8})
plt.title('Matriks Korelasi', fontsize=16)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()
```



Matriks korelasi di visualisasikan dalam bentuk heatmap dengan menggunakan library seaborn dan matplotlib. Dapat diketahui bahwa semakin cerah warnanya, semakin tinggi korelasi antara kedua kolom tersebut. Dari heatmap tersebut dapat disimpulkan korelasi tertinggi yaitu antara “Price_Range” dan “RAM” dengan korelasi 1:0.92 dan ada juga dari “3G” ke “4G” dan “Front_Camera” ke “Primary_Camera” dengan perbandingan korelasi 0.58:1 dan 0.64:1

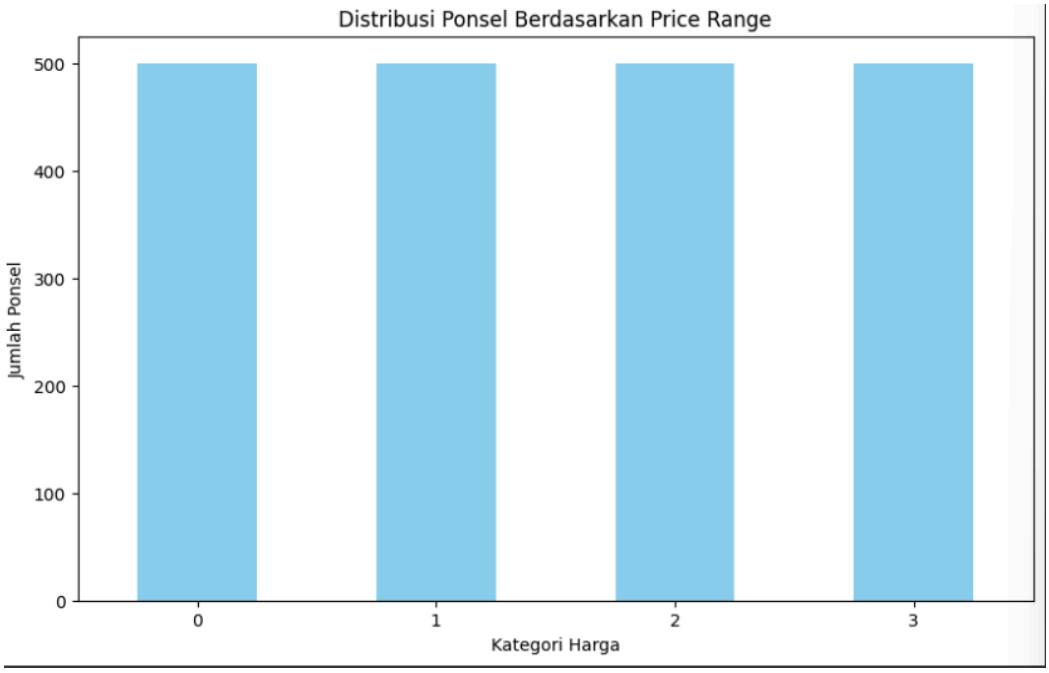
```
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidt
plt.title("Heatmap Matriks Korelasi")
plt.show()
```



Matriks korelasi di visualisasikan dalam bentuk heatmap dengan menggunakan library seaborn dan matplotlib. Dapat diketahui jika semakin mendekati merah warnanya, semakin tinggi korelasi antara kedua kolom tersebut. Dari heatmap diatas dapat disimpulkan korelasi tertinggi yaitu antara “Price_Range” dan “RAM” dengan korelasi 1:0.92 dan ada juga dari “3G” ke “4G” dan “Front_Camera” ke “Primary_Camera” dengan perbandingan korelasi 0.58:1 dan 0.64:1.

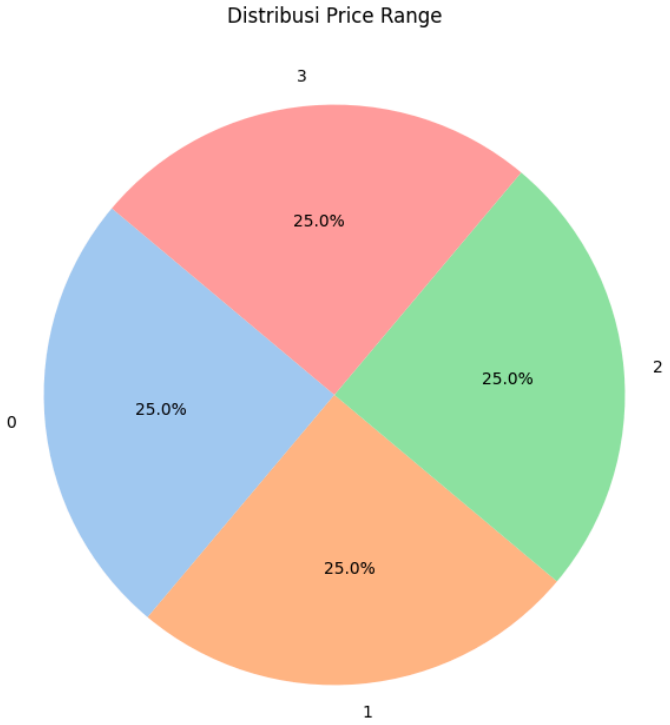
EDA

1. Bar Chart



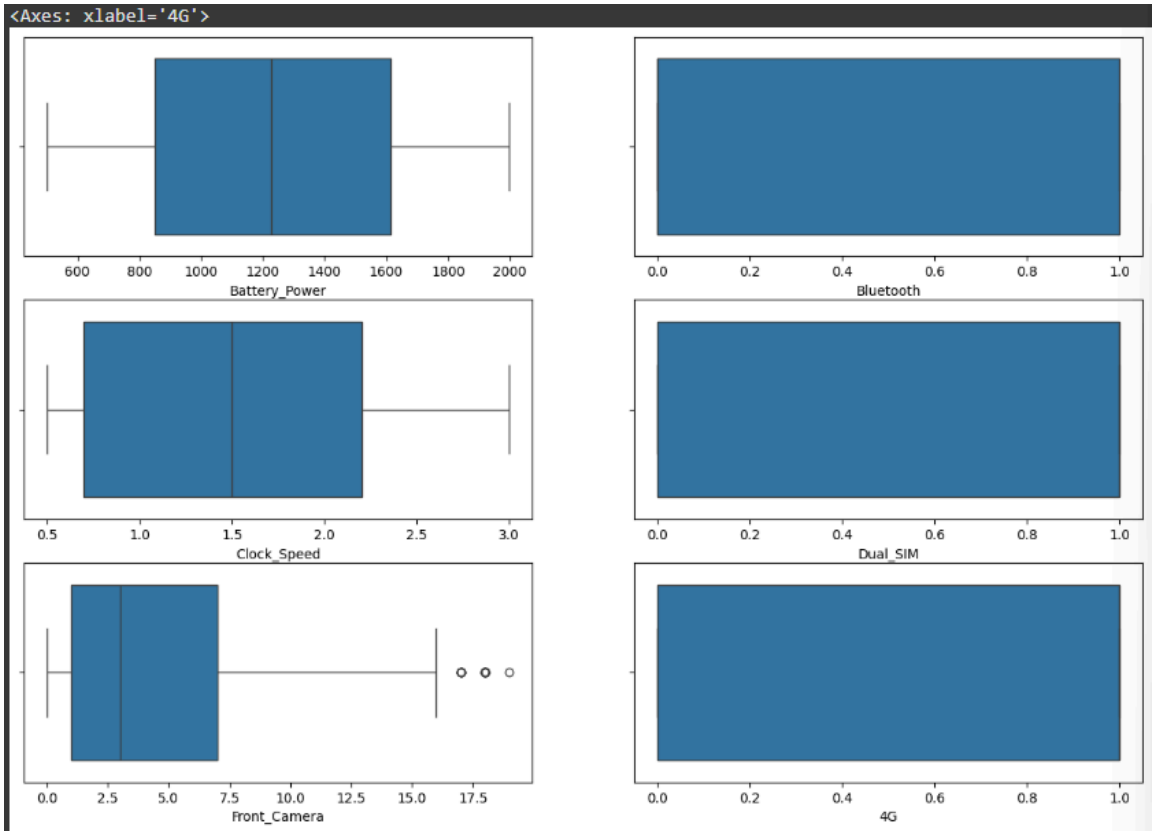
Pada bagian bar chart, membandingkan distribusi jumlah ponsel berdasarkan kategori harga (kolom “Price_Range”, sebagai catatan, kategori 0 adalah untuk *cheap* atau kategori harga murah, kategori 1 untuk *average price* atau kategori harga rata-rata, kategori 2 untuk *expensive* atau kategori harga mahal, dan kategori 3 untuk *very expensive* atau kategori harga sangat mahal. Setelah dibandingkan diketahui bahwa untuk keempat kategori tersebut masing-masing kategori memiliki sekitar 500 jumlah ponsel.

2. Pie Chart

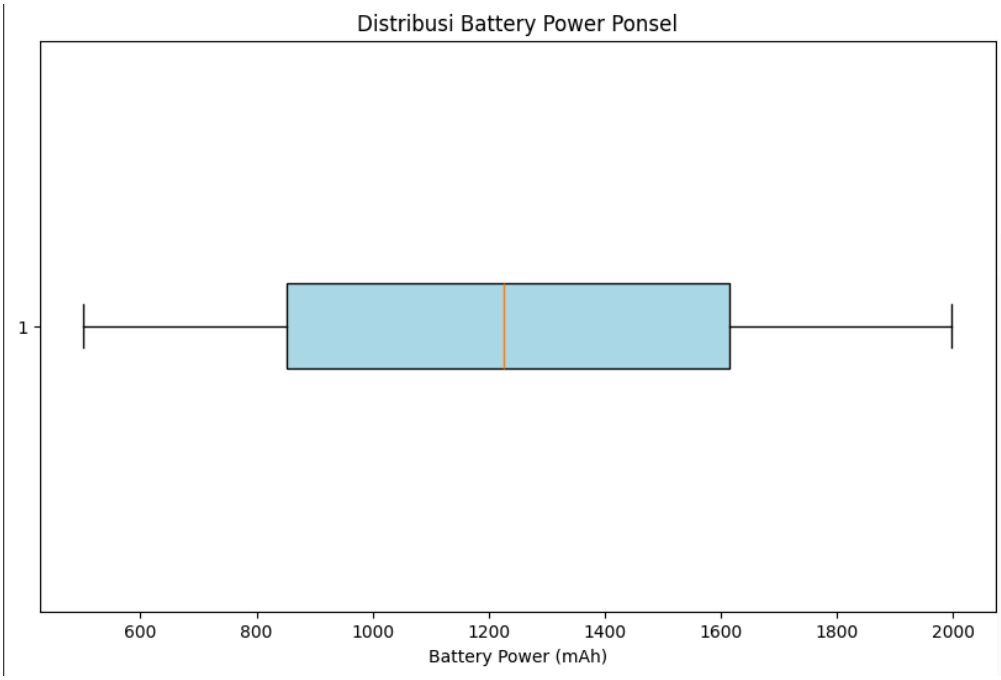


Pada bagian pie chart, membandingkan distribusi jumlah ponsel berdasarkan kategori harga (kolom “Price_Range”, sebagai catatan, kategori 0 adalah untuk *cheap* atau kategori harga murah, kategori 1 untuk *average price* atau kategori harga rata-rata, kategori 2 untuk *expensive* atau kategori harga mahal, dan kategori 3 untuk *very expensive* atau kategori harga sangat mahal. Setelah dibandingkan diketahui bahwa untuk keempat kategori tersebut masing-masing kategori terdiri atas 25% dari total jumlah ponsel.

3. **Boxplot**

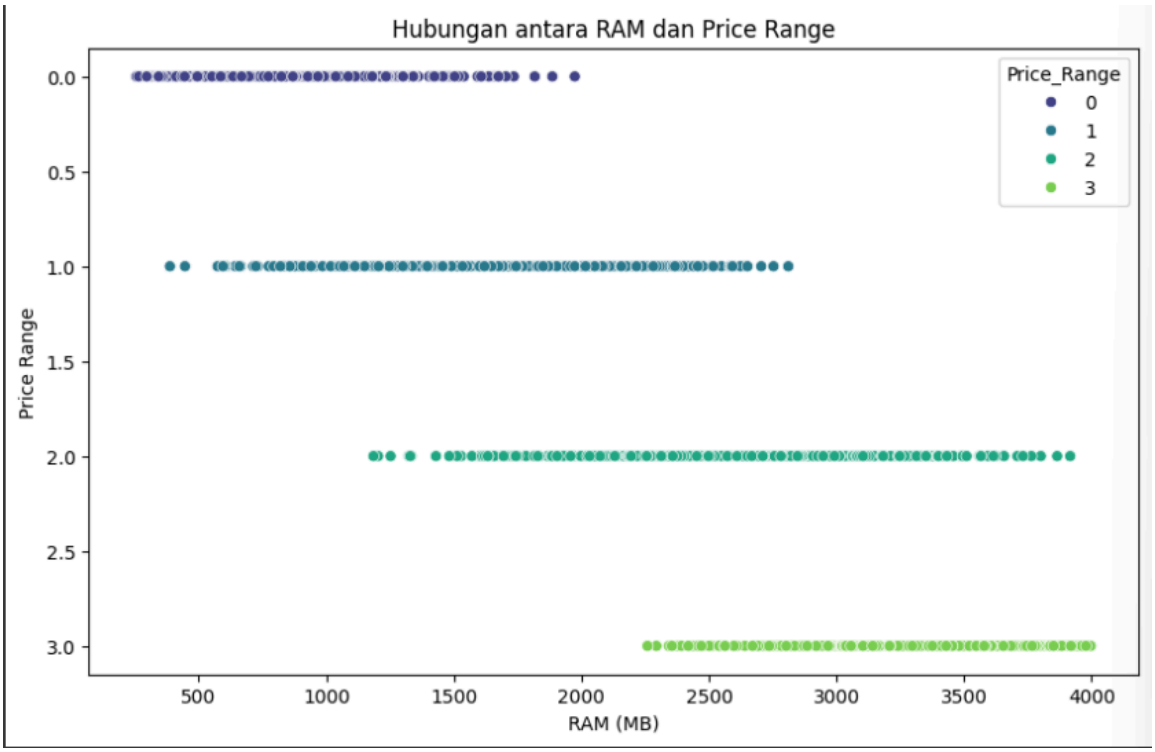


Pada bagian boxplot, menampilkan visualisasi untuk 6 fitur yang ada dalam dataset, yaitu “Battery_Power”, “Bluetooth”, “Clock_Speed”, “Dual_SIM”, “Front_Camera”, dan “4G”. Boxplot sendiri digunakan untuk menunjukkan distribusi data, nilai minimum, maksimum, kuartil pertama (Q1), median, kuartil ketiga (Q3), dan outlier. Dapat terlihat dari hasil visualisasi bahwa data dari ke-enam kolom tersebut sudah bersih dari outlier. Bagian boxplot ini ditampilkan dalam 1 figur ukuran 15x10, sehingga ke-enam boxplot akan tampil secara lebih ringkas dengan pemanggilan subplot.



Pada bagian boxplot, menampilkan boxplot horizontal menggunakan matplotlib dengan ukuran 10x6. Visualisasi ini membantu dalam memahami penyebaran data dan mengidentifikasi outlier untuk fitur kolom “Battery_Power”.

4. Scatter Plot



Pada bagian scatter plot, menunjukkan hubungan antara kolom “RAM” (kapasitas RAM dalam MB) dan “Price_Range” (kategori harga). Scatter plot dibuat menggunakan Seaborn dengan ukuran gambar 10x6, di mana sumbu x menunjukkan nilai “RAM” dan sumbu y menunjukkan “Price_Range”. Poin-poin dalam plot diberi warna berbeda sesuai nilai “Price_Range” menggunakan palet warna viridis untuk membedakan kategori harga. Berdasarkan scatter plot tersebut, dapat diketahui bahwa semakin tinggi RAM suatu ponsel, maka semakin tinggi pula harga dari ponsel tersebut.

Pemilihan fitur/kolom

Kolom yang kami gunakan untuk penelitian ini adalah battery_power, bluetooth, clock_speed, dual_sim, front_camera, 4G, internal_memory, mobile_depth, mobile_weight, num_cores, primary_camera, pixel_height, pixel_width, ram, screen_height, screen_width, talk_time, 3G, touch_screen, dan wifi. Kolom-kolom ini memiliki peran penting dalam penelitian yang kami lakukan, tanpa menggunakan kolom-kolom ini, kami tidak dapat melakukan klasifikasi ponsel sesuai dengan kategori harga yang diinginkan.

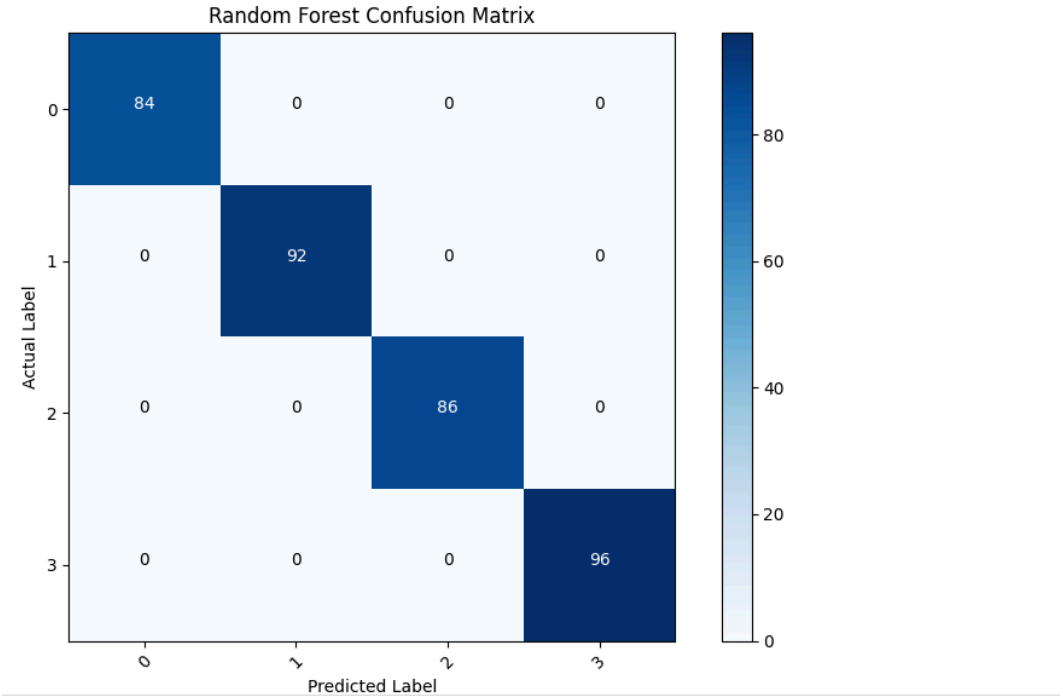
Machine Learning

1. Random Forest

Random Forest, merupakan algoritma yang digunakan untuk ensemble learning dengan membangun beberapa decision trees dan menggabungkan hasil prediksi untuk meningkatkan akurasi. Metode ini cocok untuk data klasifikasi maupun regresi karena memiliki kemampuan untuk menangani overfitting. Dengan menggunakan algoritma ini, kami mendapatkan hasil evaluasi model sebanyak 1.000.

```
Hasil Evaluasi Model Random Forest:
Akurasi: 1.0000
F1 Score: 1.0000
Presisi: 1.0000
Recall: 1.0000

Confusion Matrix:
prediction    0.0   1.0   2.0   3.0
Price_Range
0            84.0   0.0   0.0   0.0
1             0.0  92.0   0.0   0.0
2             0.0   0.0  86.0   0.0
3             0.0   0.0   0.0  96.0
```

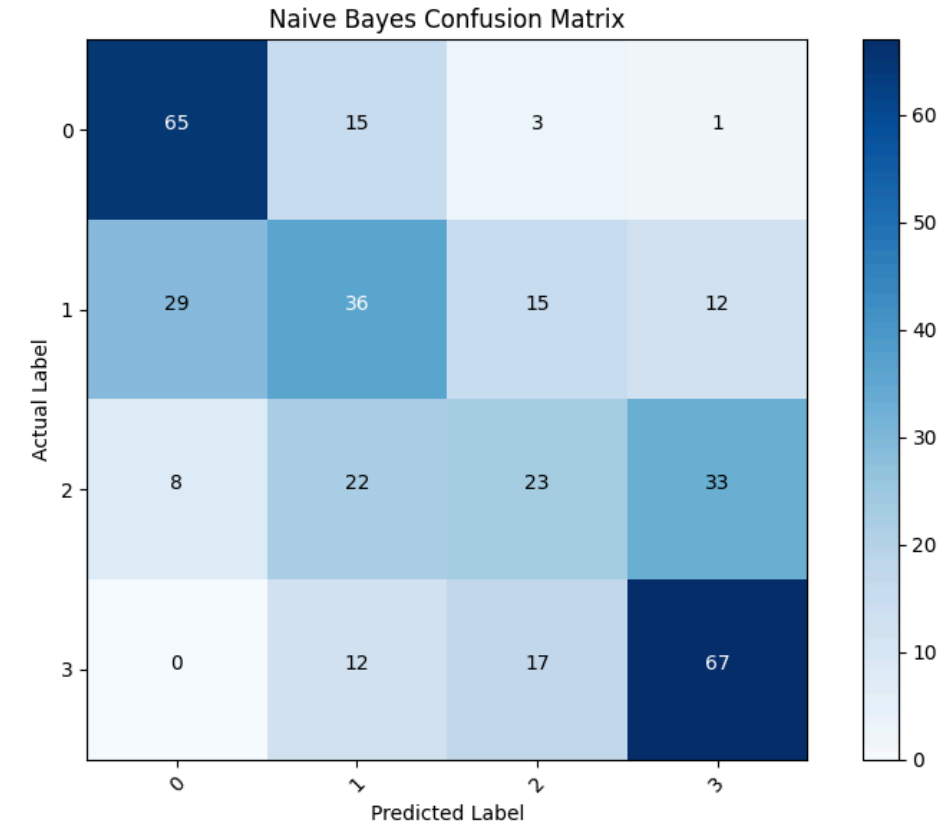


2. Naive Bayes

Algoritma Naive Bayes adalah algoritma klasifikasi probabilistik yang berdasarkan pada teorema bayes. Meskipun asumsi Naive yang digunakan (yaitu, asumsi bahwa fitur-fitur dalam suatu dataset independen satu sama lain) seringkali tidak benar dalam praktiknya, algoritma ini tetap efektif dan sering digunakan. Dengan menggunakan algoritma ini, kami mendapatkan Akurasi sebesar 53.35%, F1-Score sebesar 51,72%, presisi sebesar 51.26%, dan recall sebesar 53.35%.

```
Naive Bayes Results:
Accuracy: 0.5335
F1 Score: 0.5172
Presisi: 0.5126
Recall: 0.5335

Confusion Matrix:
prediction    0.0   1.0   2.0   3.0
Price_Range
0            65.0  15.0   3.0   1.0
1            29.0  36.0  15.0  12.0
2             8.0  22.0  23.0  33.0
3             0.0  12.0  17.0  67.0
```



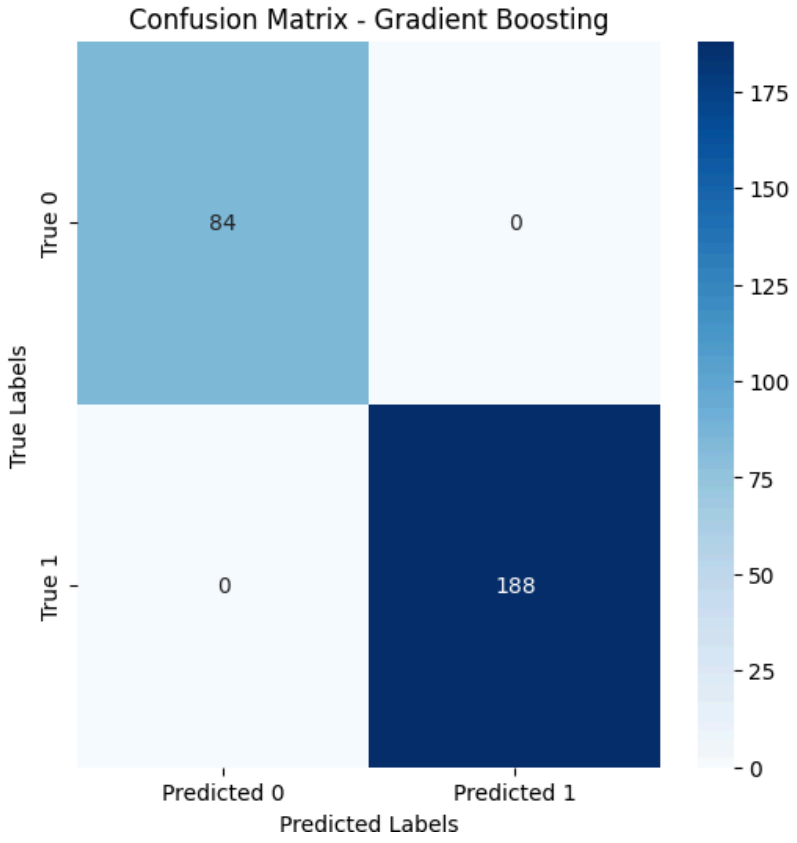
Diagonal utama menunjukkan jumlah sampel yang diprediksi dengan benar untuk masing-masing kelas:

a. Kelas 0: 65 sampel benar, 19 salah.
b. Kelas 1: 36 sampel benar, 56 salah.
c. Kelas 2: 23 sampel benar, 53 salah.
d. Kelas 3: 67 sampel benar, 29 salah.

3. Gradient Boosting

Gradient Boosting adalah algoritma ensemble yang bekerja dengan membangun model secara bertahap, di mana setiap model baru memperbaiki kesalahan prediksi dari model sebelumnya. Model ini sangat kuat untuk data dengan fitur yang kompleks.Dengan menggunakan algoritma ini, kami mendapatkan semua hasil akurasinya sebanyak 1.000

```
Hasil Evaluasi Model Gradient Boosting:
Akurasi: 1.0000
Presisi: 1.0000
Recall: 1.0000
F1 Score: 1.0000
/usr/local/lib/python3.11/dist-packages/pyspark/sql/context.py:158: FutureWarning: Deprecat
  warnings.warn(
```



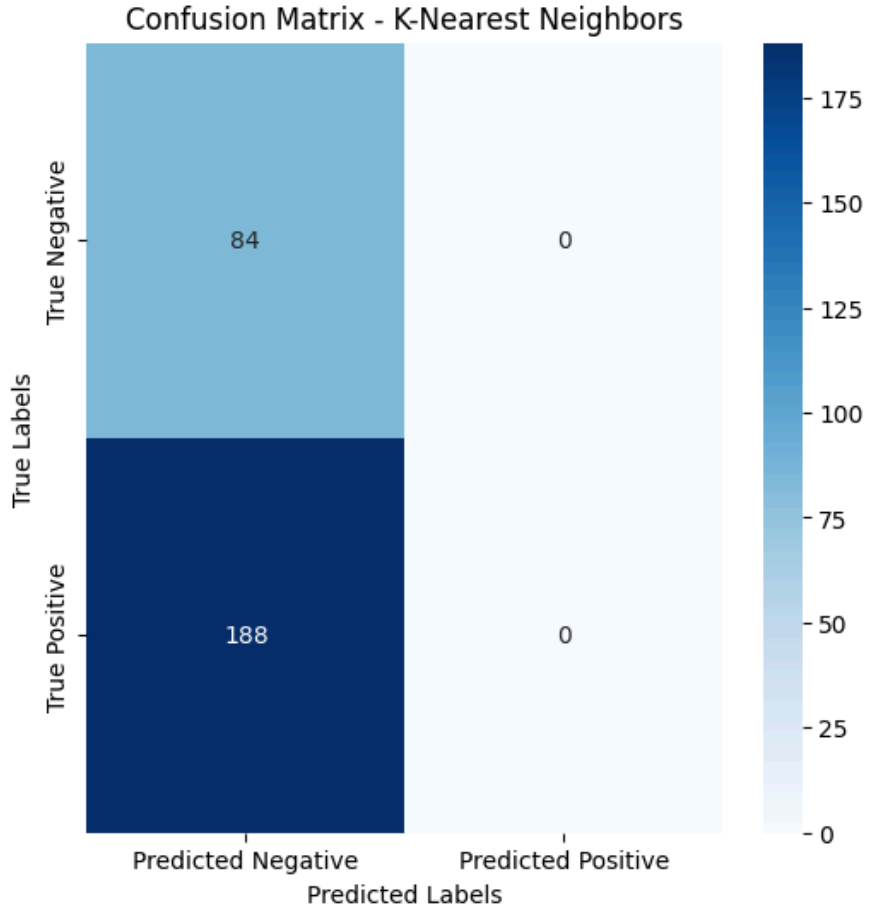
4. KNN

K-Nearest Neighbors (KNN) adalah algoritma berbasis instance learning yang menentukan kelas data baru berdasarkan mayoritas. Kelebihan dari KNN ini sendiri adalah dapat mengimplementasi dengan sederhana dan fleksibilitas untuk menangani data non-linear. Namun juga terdapat kelemahan seperti sensitif terhadap ketidakseimbangan data dan juga rentan terhadap noise. Dengan menggunakan algoritma ini, kami mendapatkan akurasi sebesar 30.88%.

```
K-Nearest Neighbors Results:
Accuracy: 0.3088
              precision    recall  f1-score   support

         0.0       0.31      1.00      0.47        84
         1.0       0.00      0.00      0.00       188

    accuracy                           0.31       272
   macro avg       0.15      0.50      0.24       272
weighted avg       0.10      0.31      0.15       272
```



Hasil Analisa

Hasil analisis menunjukkan bahwa **Random Forest** memberikan performa yang sangat baik dengan akurasi tinggi karena menggunakan pendekatan ensemble yang menggabungkan banyak decision tree untuk menghasilkan prediksi yang akurat dan stabil. **Naive Bayes**, meskipun sederhana dan cepat, memiliki performa yang kurang optimal karena asumsi independensi antar fitur yang tidak sepenuhnya sesuai dengan data yang digunakan. Selanjutnya, **Gradient Boosting** menunjukkan kemampuan yang sangat baik dalam menangani data kompleks, dengan proses iteratif yang secara bertahap memperbaiki

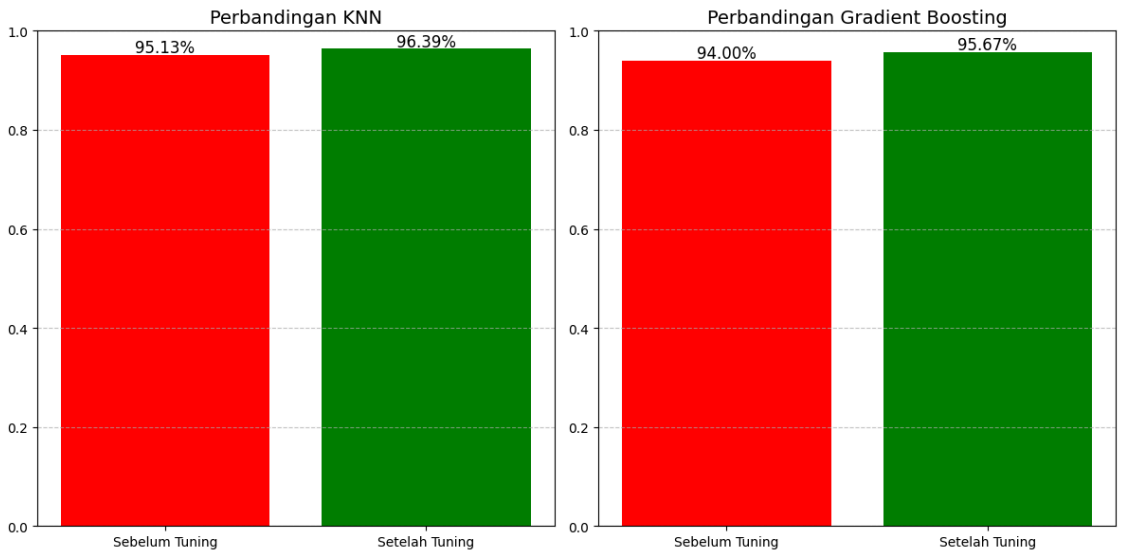
kesalahan prediksi sebelumnya. **KNN**, meskipun awalnya menunjukkan performa yang lebih rendah dibandingkan model lain, tetap mampu menghasilkan prediksi yang baik, terutama ketika digunakan pada data yang lebih sederhana.

Jadi dari keempat model yang telah kami uji, **KNN** dan **Gradient Boosting** dipilih sebagai model terbaik karena keduanya menunjukkan potensi besar dalam menghasilkan prediksi yang akurat, serta fleksibilitas untuk ditingkatkan melalui proses hyperparameter tuning yang dilakukan pada langkah selanjutnya.

Hyperparameter Tuning

Hyperparameter tuning adalah proses mencari kombinasi terbaik dari nilai hyperparameter dalam model pembelajaran mesin untuk meningkatkan performa model. Hyperparameter adalah parameter yang tidak dapat dipelajari oleh model selama proses pelatihan dan harus diatur sebelum pelatihan dimulai. Tujuan dari hyperparameter tuning adalah mencari kombinasi nilai hyperparameter yang optimal sehingga model dapat memberikan kinerja yang lebih baik pada data uji atau data baru yang belum pernah dilihat.

Hyperparameter Tuning KNN dengan Gradient Boosting



Proses hyperparameter tuning diatas dilakukan untuk meningkatkan performa model KNN dan Gradient Boosting. Pada KNN, tuning dilakukan dengan menyesuaikan jumlah tetangga terdekat (*k*), fungsi jarak (*distance metric*), dan metode pembobotan (*weighting*).

Nilai optimal untuk *k* ditemukan pada rentang menengah (7 hingga 11), yang memberikan keseimbangan antara bias dan variansi, sehingga meningkatkan akurasi. Namun, meskipun performa KNN meningkat, model ini tetap sensitif terhadap noise dan kurang efektif pada data yang kompleks.

Untuk Gradient Boosting, tuning melibatkan parameter seperti *learning rate*, jumlah estimators (*n_estimators*), dan maksimum kedalaman (*max_depth*). Kombinasi terbaik ditemukan dengan *learning rate* rendah (0.01–0.1), jumlah estimators sedang hingga tinggi (100–200), dan *max_depth* yang disesuaikan untuk mencegah overfitting. Hasil tuning menunjukkan bahwa Gradient Boosting secara konsisten memberikan performa yang lebih baik dibandingkan KNN, terutama untuk data dengan kompleksitas tinggi.

Kesimpulan

Melalui tahapan preprocessing, data berhasil dibersihkan, diolah, dan disiapkan untuk analisis, termasuk penanganan tipe data, penghapusan nilai null, serta eksplorasi hubungan antar fitur melalui visualisasi seperti heatmap, boxplot, dan scatter plot. Dari analisis korelasi, ditemukan bahwa fitur RAM memiliki hubungan yang sangat kuat dengan kategori harga ponsel (*price range*), sehingga menjadi faktor utama dalam prediksi.

Pada tahapan implementasi algoritma, kami menggunakan empat model yaitu: **Random Forest**, **Naive Bayes**, **Gradient Boosting**, dan **KNN**. KNN dan Gradient Boosting memberikan performa terbaik karena mampu menangani data kompleks dan melakukan generalisasi dengan baik. Random Forest memberikan hasil yang cukup baik untuk model sederhana, meskipun kurang optimal pada data dengan hubungan antar fitur yang kompleks. Sementara itu, Naive Bayes menunjukkan performa yang paling rendah karena sensitif terhadap noise dan ketidakseimbangan data.

Proses **hyperparameter tuning** berhasil meningkatkan performa KNN dan Gradient Boosting. Pada KNN, penyesuaian parameter *k* dan fungsi jarak membantu meningkatkan akurasi, meskipun model ini masih rentan terhadap kelemahan dasarnya. Pada Gradient Boosting, penyesuaian *learning rate*, jumlah estimators, dan *max depth* memberikan hasil akurasi yang optimal dan konsisten.

Jadi dapat disimpulkan bahwa KNN dan Gradient Boosting menjadi pilihan model terbaik untuk prediksi kelas harga ponsel berdasarkan spesifikasi teknisnya. Pemanfaatan metode preprocessing dan tuning yang tepat menunjukkan bahwa analisis data yang sistematis dapat menghasilkan model prediksi yang akurat.

Lampiran

Link google colab : https://linksy.site/BDPAL(pyspark)
Link dataset :
https://www.kaggle.com/datasets/jacksondivakarr/phone-classification-dataset
Link github : https://github.com/RachmasariAR/KlasifikasiHargaPonsel
Link launchinpad :
https://www.launchinpad.com/project/phone-price-classification-7b27c1c