BAB III

HASIL ANALISIS DAN PEMBAHASAN

3.1 Persiapan Data

Persiapan data adalah langkah penting untuk memastikan data siap digunakan dalam model analisis atau *machine learning*. Dalam konteks penelitian untuk menganalisis kepuasan pelanggan menggunakan algoritma *Naïve Bayes* pada program Penyediaan Air Minum dan Sanitasi Berbasis Masyarakat (PAMSIMAS) di Desa Batuah Kecamatan Loa Janan Kabupaten Kutai Kartanegara, proses *preprocessing* data melibatkan beberapa tahapan sebagai berikut :

3.1.1 Mengumpulkan Data

Data dikumpulkan dari kuesioner yang telah diisi oleh responden. Data ini mencakup berbagai atribut seperti jenis kelamin**,** pekerjaan, pendidikan, kualitas air: kejernihan, rasa dan bau air, kuantitas: ketersediaan air di pelanggan, kontinuitas: pengaliran air selama 24 jam, pelayanan: kecepatan, keramahan, dan ketepatan waktu pelayanan, penanganan: kecepatan, ketepatan dan penyelesaian keluhan.

3.1.2 Memuat Data

```
# Import libraries
import pandas as pd
from google.colab import files
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

Gambar 1.1 Kode Python Untuk Import Library

Pada Gambar 3.1 dijelaskan bahwa data yang dikumpulkan dimuat kedalam program menggunakan Pustaka seperti ***import pandas as pd*** untuk memanipulasi dan menganalisis data

yang berbentuk tabel. pandas menyediakan struktur data yang sangat fleksibel untuk mengolah data.

Perintah ***from google.colab import files*** untuk mengunggah atau mengunduh file dari Google Colab. Modul ini berguna untuk bekerja dengan data yang berada di perangkat lokal.

Perintah ***from sklearn.model_selection import train_test_split*** untuk membagi dataset menjadi dua bagian: data latih dan data uji. Pembagian ini penting untuk mengevaluasi kinerja model secara objektif.

Perintah ***from sklearn.naive_bayes import GaussianNB*** untuk membuat model *Naïve Bayes* dengan distribusi *Gaussian*. Algoritma ini digunakan untuk klasifikasi berdasarkan probabilitas.

Perintah ***from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix*** untuk menghitung metrik evaluasi kinerja model: *accuracy_score*: Menghitung akurasi prediksi. *precision_score*: Menghitung presisi prediksi. *recall_score*: Menghitung *recall* prediksi. *f1_score*: Menghitung *F1-score*, gabungan dari presisi dan *recall*. *confusion_matrix*: Menghasilkan matriks kebingungan untuk melihat performa klasifikasi.

Perintah ***import matplotlib.pyplot as plt*** digunakan untuk membuat visualisasi data seperti grafik dan plot.

Perintah ***import seaborn as sns*** digunakan untuk meningkatkan kemampuan visualisasi data yang ditawarkan oleh matplotlib. seaborn digunakan untuk membuat grafik yang lebih menarik dan informatif.

```
[ ]  # Unggah file
     uploaded = files.upload()
```

Gambar 3.2 Import Data di Google Collab

Kode pada Gambar 3.2 digunakan untuk mengunggah file di lingkungan Google Colab dengan menggunakan library files dari modul google.colab. ***files***: Merupakan modul yang disediakan oleh Google Colab untuk mengelola pengunggahan file**, *upload*()**: Metode *upload*() dari files digunakan untuk menampilkan dialog pengunggahan file interaktif, ketika kode dieksekusi di Google Colab, maka akan memunculkan kotak dialog untuk memilih file dari sistem lokal dan mengunggahnya ke sesi notebook, ***uploaded***: Variabel *uploaded* digunakan untuk menyimpan hasil dari proses pengunggahan file. Secara khusus, ini akan berisi data file yang diunggah dalam bentuk *dictionary*, di mana kunci adalah nama file dan nilai adalah konten dari file tersebut.

```
# Memuat data
file_name = list(uploaded.keys())[0]
data = pd.read_excel('kuesioner skripsii.xlsx')

# menampilkan data
data.head()
```

	No	Jenis Kelamin	Pekerjaan	Pendidikan Terakhir	Kualitas Kejernihan Air	Kualitas Rasa Air	Kualitas Bau Air	Kuantitas Ketersediaan Air	Kontinuitas Pengaliran Air	Kecepatan Pelayanan	Keramahan Pelayanan	Ketepatan Waktu Pelayanan	Kecepatan Tanggapan Keluhan	Ketepatan Tanggapan Keluhan	Penyelesaian Keluhan	Jumlah	Rata - Rata	class Kepuasan
0	1	2	1	3	4	5	5	5	4	5	5	5	5	5	5	53	4.818182	Puas
1	2	1	3	6	4	4	4	4	3	4	4	4	3	3	3	40	3.636364	Puas
2	3	2	1	5	4	3	5	4	4	4	5	4	3	3	3	42	3.818182	Puas
3	4	2	1	3	5	5	5	4	4	4	4	4	4	4	4	47	4.272727	Puas
4	5	2	1	3	4	4	4	5	4	5	5	5	5	5	5	51	4.636364	Puas

Gambar 3.3 Memuat dan Menampilkan Data

Kode pada Gambar 3.3 merupakan dua pernyataan python yang digunakan untuk memuat dan menampilkan data. Perintah ***file_name* = *list(uploaded.keys())*[0]** bertujuan untuk mendapatkan nama file yang diunggah dari variabel uploaded yang telah dijelaskan sebelumnya, ***uploaded*** adalah dictionary yang berisi hasil dari fungsi, ***files.upload*()**, di mana kunci adalah nama file dan nilai adalah konten dari file yang diunggah, ***uploaded.keys*()** mengembalikan semua kunci (nama file) dari dictionary ***uploaded***, ***list(uploaded.keys())*** mengubah kunci-kunci ini menjadi daftar, **[0]** mengambil kunci pertama dari daftar tersebut, sehingga ***file_name*** akan berisi nama file pertama yang diunggah ke dalam sesi notebook.

Perintah **data = *pd.read_excel*('kuesioner skripsii.xlsx')** Pernyataan ini menggunakan pandas (dengan alias pd) untuk membaca file Excel dengan nama 'kuesioner skripsii.xlsx' dan menyimpannya ke dalam variabel data, **pd** adalah alias untuk pandas, yang merupakan library Python yang kuat untuk analisis data, ***pd.read_excel*('kuesioner skripsii.xlsx')** adalah fungsi dari pandas yang membaca file Excel ('.xlsx') yang disebut 'kuesioner skripsii.xlsx' dari sistem file lokal atau dari lokasi yang sudah ditentukan, hasil dari fungsi ***pd.read_excel*()** disimpan dalam variabel data, yang kemungkinan besar berupa DataFrame pandas. DataFrame adalah struktur data tabular dua dimensi yang sangat berguna untuk memanipulasi dan menganalisis data.

Perintah ***data.head*()**: berfungsi untuk menampilkan lima baris pertama dari DataFrame secara default.

3.1.3 Memeriksa Data

```
# Memeriksa Data

print(data.describe())

print(data.info())
```

Gambar 3.2 Memeriksa Data

Kode pada Gambar 3.4 digunakan untuk memeriksa data yang dimuat untuk memastikan bahwa semua data telah diimpor dengan benar dan memahami struktur data. Ini termasuk melihat beberapa baris pertama dari dataset dan mendapatkan ringkasan statistik. ***print(data.head())*** digunakan untuk menampilkan lima baris pertama dari *DataFrame* data untuk mendapatkan gambaran awal tentang struktur dan isi data.

Perintah ***print(data.describe())*** digunakan untuk menyediakan statistik deskriptif dari kolom-kolom numerik dalam *DataFrame* data. Ini membantu dalam memahami distribusi dan karakteristik dasar dari data. ***data.describe*()**: Fungsi ini memberikan ringkasan statistik

dari kolom-kolom numerik, termasuk count (jumlah non-NaN entri), mean (rata-rata), std (standar deviasi), min (nilai minimum), 25% (kuartil pertama), 50% (median atau kuartil kedua), 75% (kuartil ketiga), dan max (nilai maksimum), ***print(data.describe())***: Menampilkan hasil data.describe() ke layar.

Perintah **print(data.info())** digunakan untuk menampilkan informasi umum tentang DataFrame data, termasuk jumlah total entri, jumlah non-NaN entri di setiap kolom, tipe data dari setiap kolom, dan penggunaan memori. **data.info()**: Fungsi ini memberikan informasi ringkas tentang DataFrame, termasuk indeks, tipe data, jumlah nilai non-NaN di setiap kolom, dan penggunaan memori, **print(data.info())**: Menampilkan hasil data.info() ke layar.

3.1.4 Mengkonversi Data Kategorikal ke Numerik

```
[4]  # Preprocessing data (sesuaikan dengan struktur data Anda)
     # Contoh konversi data kategorikal ke numerik
     data['class Kepuasan'] = data['class Kepuasan'].map({'Tidak Puas': 1, 'Puas': 2})
```

Gambar 3.3 Mengkonversi Data Kategori ke Numerik

Kode pada Gambar 3.5 berfungsi untuk melakukan *preprocessing* data, khususnya mengonversi data kategorikal menjadi data numerik. Algoritma *Naïve Bayes* memerlukan data numerik, oleh karena itu, data kategorikal seperti *class* kepuasan perlu dikonversi menjadi data numerik. **data[*'class* Kepuasan']**: merujuk pada kolom bernama "*class* Kepuasan" dalam DataFrame **data**. Kolom ini berisi data kategorikal yang mencakup nilai-nilai seperti 'Tidak Puas' dan 'Puas'. Perintah **.map({'Tidak Puas': 1, 'Puas': 2})**: Metode **.map()** yang digunakan untuk menggantikan setiap nilai dalam kolom "class Kepuasan" dengan nilai baru berdasarkan mapping yang diberikan. *Dictionary* **{'Tidak Puas': 1, 'Puas': 2}** untuk menentukan bahwa nilai 'Tidak Puas' akan digantikan dengan 1 dan nilai 'Puas' akan digantikan dengan 2.

3.1.5 Membersihkan Data

```
# Mengisi missing values dengan metode yang sesuai (misalnya median, mean, mode, atau nilai tertentu)
data.fillna(data.median(), inplace=True)

# Menghapus duplikasi
data.drop_duplicates(inplace=True)
```

Gambar 3.4 Membersihkan Data

Kode pada Gambar 3.6 terdiri dari dua bagian: Perintah ***data.fillna(data.median(), inplace=True)*** digunakan untuk mengisi nilai yang hilang (*missing values*) di dalam DataFrame **data** dengan nilai median dari setiap kolom, ***data.fillna()***: Fungsi ini digunakan untuk mengisi nilai yang hilang (NaN) di dalam DataFrame, ***data.median()***: Metode ini menghitung nilai median untuk setiap kolom dalam DataFrame data, nilai median adalah nilai tengah dalam kumpulan data yang telah diurutkan, ***inplace=True***: Argumen ini memastikan bahwa pengisian nilai yang hilang dilakukan langsung pada DataFrame **data** tanpa perlu membuat salinan baru. Dengan kode ini, setiap nilai yang hilang di setiap kolom akan digantikan dengan nilai median dari kolom tersebut.

Perintah ***data.drop_duplicates(inplace=True)*** digunakan untuk menghapus baris-baris duplikat dari DataFrame data, ***data.drop_duplicates()***: Fungsi ini menghapus baris-baris yang memiliki data yang sama persis dengan baris lain dalam DataFrame, ***inplace=True***: Argumen ini memastikan bahwa penghapusan baris duplikat dilakukan langsung pada DataFrame data tanpa perlu membuat salinan baru. Dengan kode ini, hanya baris-baris unik yang akan tetap ada dalam DataFrame data, sementara baris-baris yang merupakan duplikasi akan dihapus.

Secara keseluruhan, kode ini pertama-tama mengisi nilai yang hilang dalam DataFrame data dengan nilai median dari setiap kolom, lalu menghapus baris-baris yang duplikat untuk memastikan bahwa DataFrame tersebut hanya mengandung data yang unik.

3.2 Pembagian Data

1.2.1 Membagai Data Menjadi Fitur dan Label

```
[5] # Memisahkan fitur dan label
    X = data.drop('class Kepuasan', axis=1)
    y = data['class Kepuasan']
```

Gambar 3.5 Membagi Data Menjadi Fitur dan Label

Kode pada Gambar 3.7 digunakan untuk memisahkan dataset menjadi dua bagian: fitur (*independent variable*) dan label (*dependent variable*). Proses ini penting karena dalam *machine learning*, model dilatih menggunakan fitur untuk memprediksi label. Perintah ***data.drop('class Kepuasan', axis=1)*** digunakan untuk menghapus kolom 'class Kepuasan' dari DataFrame data, ***axis*=1** menunjukkan bahwa penghapusan dilakukan pada kolom, bukan baris, hasilnya adalah DataFrame **X** yang berisi semua kolom kecuali kolom 'class Kepuasan'. Kolom-kolom ini adalah fitur yang akan digunakan untuk melatih model.

Perintah ***data['class Kepuasan']*** digunakan untuk memilih kolom *'class* Kepuasan' dari DataFrame **data**, hasilnya adalah Series y yang berisi label atau target yang akan diprediksi oleh model.

1.2.2 Membagi Data Menjadi Data Latih dan Data Uji

```
[6] # Membagi data menjadi data latih dan data uji
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Gambar 3.6 Membagi Data Menjadi Data Latih dan Data Uji

Kode pada Gambar 3.8 digunakan untuk membagi dataset menjadi dua bagian: data latih (*training data*) dan data uji (*testing data*). Proses ini penting dalam machine learning untuk melatih model dengan satu bagian data dan menguji performanya dengan bagian data lainnya. Fungsi ***train_test_split*** dari pustaka ***sklearn.model_selection*** digunakan untuk membagi dataset menjadi dua bagian: data latih dan data uji, **X** adalah fitur (independent variables) dari dataset, y adalah label atau target (dependent variable) dari dataset, ***test_size*=0.2** menunjukkan bahwa 20% dari data akan digunakan sebagai data uji, sementara 80% sisanya akan digunakan sebagai data latih, ***random_state*=42** adalah parameter yang digunakan untuk memastikan

bahwa pembagian data bersifat deterministik dan dapat direproduksi. Ini berarti setiap kali kode dijalankan dengan random_state yang sama, pembagian data akan menghasilkan bagian yang sama. Hasil Pembagian: ***X_train***: Data fitur untuk pelatihan model, ***X_test***: Data fitur untuk pengujian model, ***y_train***: Data label untuk pelatihan model, ***y_test***: Data label untuk pengujian model.

3.3 Pelatihan Model

3.3.1 Melatih Model Naïve Bayes

```
# Melatih model Naive Bayes
model = GaussianNB()
model.fit(X_train, y_train)
```

Gambar 3.7 Melatih Model Naïve Bayes

Kode pada Gambar 3.9 digunakan untuk melatih model *Naive Bayes* menggunakan algoritma *Gaussian Naïve Bayes* (*GaussianNB*) dengan data yang telah diproses dan dibagi sebelumnya. ***GaussianNB*** digunakan untuk membuat model *Gaussian Naïve Bayes*. **Model = *GaussianNB*()** untuk membuat sebuah instance dari model *Gaussian Naïve Bayes*, pada titik ini model masih kosong dan belum dilatih dengan data apapun.

Perintah ***fit*()** adalah metode yang digunakan untuk melatih model dengan data yang telah dibagi menjadi data latih (X_train dan y_train). ***X_train*** merupakan data fitur (*variabel independen*) yang digunakan untuk melatih model. ***y_train*** merupakan label atau target (*variabel dependen*) yang sesuia dengan fitur tersebut.

3.4 Pengujian Model

3.4.1 Memprediksi Data Uji

```
[8]  # Memprediksi data uji
     y_pred = model.predict(X_test)
```

Gambar 3.8 Memprediksi Data Uji

Kode pada Gambar 3.10 bertujuan untuk memprediksi nilai-niai dari data uji menggunakan model yang telah dilatih sebelumnya. ***model.predict*()** adalah metode yang digunakan untuk memprediksi nilai label atau target berdasarkan model yang telah dilatih**.** ***X_test*** adalah data fitur (*variabel independen*) dari set uji yang akan diprediksi oleh model. ***y_pred*** digunakan untuk menyimpan hasil prediksi model untuk *X_test*.

3.5 Evaluasi

3.5.1 Menghitung Metrik Evaluasi

```
# Evaluasi model
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
```

Gambar 3.9 Menghitung Metrik Evaluasi

Kode pada Gambar 3.11 digunakan untuk menghitung berbagai metrik evaluasi untuk mengukur kinerja model *Naïve Bayes* yang telah dilatih. Metrik evaluasi membantu dalam memahami seberapa baik model melakukan prediksi berdasarkan data uji. Perintah ***confusion_matrix(y_test, y_pred)*** digunakan untuk menghasilkan matriks kebingungan, yang menunjukkan perbandingan antara prediksi model dan nilai sebenarnya, Matriks kebingungan membantu dalam melihat jumlah prediksi benar dan salah untuk masing-masing kelas.

Perintah ***accuracy_score(y_test, y_pred)*** digunakan untuk menghitung persentase prediksi yang benar dari total prediksi yang dibuat oleh model, akurasi merupakan rasio dari jumlah prediksi benar terhadap total jumlah prediksi.

Perintah ***precission_score(y_test, y_pred)*** digunakan untuk menghitung presisi model, yaitu rasio dari jumlah prediksi benar positif terhadap total jumlah prediksi positif, presisi mengukur ketepatan dari prediksi positif model, yaitu seberapa banyak prediksi positif yang

benar-benar positif. Dengan parameter ***average='weighted'***, nilai presisi dihitung untuk setiap kelas dan kemudian dihitung rata-rata tertimbang berdasarkan jumlah sampel untuk setiap kelas.

Perintah ***recall_score(y_test, y_pred)*** digunakan untuk menghitung recall model, yaitu rasio dari jumlah prediksi benar positif terhadap total jumlah sebenarnya positif, recall mengukur sensitivitas atau kemampuan model dalam menemukan semua sampel positif yang ada. Dengan parameter ***average='weighted'***, nilai *recall* dihitung untuk setiap kelas dan kemudian dihitung rata-rata tertimbang berdasarkan jumlah sampel untuk setiap kelas.

Perintah ***f1_score(y_test, y_pred)*** digunakan untuk menghitung *F1-score*, yaitu rata-rata harmonis dari presisi dan *recall*, *F1-score* memberikan keseimbangan antara presisi dan *recall*, berguna terutama ketika ada ketidakseimbangan kelas. Dengan parameter ***average='weighted'***, nilai *F1-Score* dihitung untuk setiap kelas dan kemudian dihitung rata-rata tertimbang berdasarkan jumlah sampel untuk setiap kelas.

3.5.2 Menampilkan Hasil Evaluasi

```
# Menampilkan Hasil Evaluasi
print(f'Confusion Matrix:\n{cm}')
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

```
Confusion Matrix:
[[ 1  1]
 [ 1 17]]
Accuracy: 0.9
Precision: 0.9
Recall: 0.9
F1 Score: 0.9
```

Gambar 3.10 Menampilkan Hasil Evaluasi

Kode pada Gambar 3.12 digunakan untuk menampilkan hasil evaluasi kinerja model *Naive Bayes* setelah dilakukan prediksi terhadap data uji. Hasil evaluasi ini berupa beberapa metrik penting seperti *accuracy, precision, recall*, *F1-score*, dan matriks kebingungan

(*confusion matrix*). Perintah ***print(f'Confusion Matrix:\n{cm}')*** digunakan untuk mencetak matriks kebingungan (*confusion matrix*) dari model, Matriks kebingungan menunjukkan perbandingan antara prediksi model dan nilai sebenarnya untuk setiap kelas.

Perintah ***print(f'Accuracy: {accuracy}')*** digunakan untuk mencetak nilai akurasi dari model, Akurasi menunjukkan persentase prediksi yang benar dari total prediksi yang dibuat oleh model.

Perintah ***print(f'Precision: {precision}')*** digunakan untuk mencetak nilai presisi dari model, Presisi adalah rasio dari jumlah prediksi benar positif terhadap total jumlah prediksi positif, mengukur ketepatan prediksi positif model.

Perintah ***print(f'Recall: {recall}')*** digunakan untuk mencetak nilai recall dari model, *Recall* adalah rasio dari jumlah prediksi benar positif terhadap total jumlah sebenarnya positif, mengukur kemampuan model dalam menemukan semua sampel positif yang ada.

Perintah ***print(f'F1 Score: {f1}')*** digunakan untuk mencetak nilai *F1-score* dari model, *F1-score* adalah rata-rata harmonis dari presisi dan *recall*, memberikan keseimbangan antara keduanya.

3.6 Hasil Analisis Kepuasan

3.6.1 Menghitung Jumlah Kemunculan Setiap Kategori

```
[32] # Misalkan kolom 'kepuasan' menyimpan data kepuasan pelanggan dengan nilai 'Puas' atau 'Tidak Puas'
     class_Kepuasan_counts = data['class Kepuasan'].value_counts()
```

Gambar 3.11 Menghitung Jumlah Kemunculan Setiap Kategori

Kode pada Gambar 3.13 digunakan untuk menghitung jumlah kemunculan (frekuensi) dari setiap kategori dalam kolom *'class* Kepuasan' pada dataset. Kode ini sangat berguna untuk memahami distribusi data kepuasan pelanggan dengan nilai 'Puas' atau 'Tidak Puas'. Perintah **data['*class* Kepuasan']** digunakan untuk mengakses kolom *'class* Kepuasan' dari dataframe

data, kolom ini menyimpan data kepuasan pelanggan yang bisa memiliki nilai 'Puas' atau 'Tidak Puas'. ***.value_counts()*** adalah metode dari pandas yang menghitung jumlah kemunculan unik dari setiap nilai dalam kolom yang diakses, hasil dari ***data['class Kepuasan'].value_counts()*** adalah sebuah Series yang berisi jumlah kemunculan masing-masing nilai unik dalam kolom *'class* Kepuasan'.

3.6.2 Membuat Diagram Batang Tingkat Kepuasan

```
# Membuat diagram tingkat kepuasan
sns.countplot(x='class Kepuasan', data=data)
plt.title('Tingkat Kepuasan Pelanggan')
plt.xlabel('Kepuasan')
plt.ylabel('Jumlah')
plt.xticks(ticks=[0, 1], labels=['Tidak Puas', 'Puas'])
plt.show()
```

Gambar 3.12 Membuat Diagram Batang Tingkat Kepuasan

Kode pada Gambar 3.14 digunakan untuk membuat diagram batang (*bar plot*) yang menunjukkan distribusi tingkat kepuasan pelanggan berdasarkan data yang ada. Kode ini menggunakan seaborn (sns), yang merupakan pustaka visualisasi data di *Python*, serta *matplotlib* untuk menampilkan *plot*. Perintah ***sns.countplot*** digunakan untuk membuat diagram batang yang menunjukkan jumlah observasi untuk setiap kategori dalam variabel yang ditentukan, perintah **x=*'class* Kepuasan'** digunakan untuk menentukan variabel pada sumbu x, dalam hal ini adalah kolom *'class* Kepuasan' dari dataset data, perintah **data=data** untuk menentukan dataset yang digunakan untuk membuat plot. Perintah ***plt.title*** digunakan untuk menetapkan judul dari *plot* menjadi 'Tingkat Kepuasan Pelanggan'. Perintah ***plt.xlabel*** digunakan untuk menetapkan label sumbu x menjadi 'Kepuasan'. Perintah ***plt.ylabel*** digunakan untuk menetapkan label sumbu y menjadi 'Jumlah'. Perintah ***plt.xticks*** digunakan untuk menetapkan label pada titik-titik sumbu x, ***ticks*=[0, 1]** untuk menetapkan posisi tick pada sumbu x, ***labels*=['Tidak Puas', 'Puas']** untuk menetapkan label yang akan ditampilkan pada tick-tick tersebut. Perintah ***plt.show*()** digunakan untuk menampilkan plot.

Gambar 3.13 Diagram Batang Tingkat Kepuasan Pelanggan

Gambar 3.15 merupakan diagram batang yang menggambarkan hasil survei terkait tingkat kepuasan pelanggan. Dari hasil survei, ditemukan bahwa jumlah pelanggan yang merasa "Tidak Puas" adalah sekitar 6 pelanggan dan jumlah pelanggan yang merasa "Puas" adalah sekitar 94 pelanggan. Diagram batang ini menunjukkan jumlah pelanggan dalam dua kategori kepuasan, yaitu "Tidak Puas" dan "Puas". Jumlah pelanggan yang puas jauh lebih banyak dibandingkan dengan pelanggan yang tidak puas, menunjukkan mayoritas pelanggan memiliki pengalaman positif dengan produk/layanan yang diberikan.

3.6.3 Menghitung Persentase Kepuasan dan Ketidakpuasan

```
# Menghitung persentase kepuasan dan tidak kepuasan
kepuasan_percentage = (class_Kepuasan_counts / class_Kepuasan_counts.sum()) * 100
```

Gambar 3.14 Menghitung Persentase Kepuasan dan Ketidakpuasan

Kode pada Gambar 3.16 digunakan untuk menghitung persentase masing-masing kategori dalam kolom *'class* Kepuasan'. Ini membantu untuk memahami proporsi data yang termasuk dalam setiap kategori, yaitu 'Puas' dan 'Tidak Puas', dalam bentuk persentase dari total data. ***class*_Kepuasan_*counts*** adalah hasil dari **data[*'class* Kepuasan'].*value_counts*()**,

yang berisi jumlah kemunculan dari masing-masing kategori ('Puas' dan 'Tidak Puas') dalam kolom 'class Kepuasan'. Perintah ***class*_Kepuasan_*counts.sum*()** untuk menghitung total jumlah data dengan menjumlahkan semua nilai frekuensi dalam ***class*_Kepuasan_*counts***. Perintah ***class*_Kepuasan_*counts* / *class*_Kepuasan_*counts.sum*()** untuk menghitung proporsi (rasio) masing-masing kategori terhadap total jumlah data, perintah **(*class*_Kepuasan_*counts* / *class*_Kepuasan_*counts.sum*()) * 100** untuk mengubah proporsi tersebut menjadi persentase dengan mengalikannya dengan 100.

3.6.4 Membuat Diagram Lingkaran untuk Persentase Tingkat Kepuasan

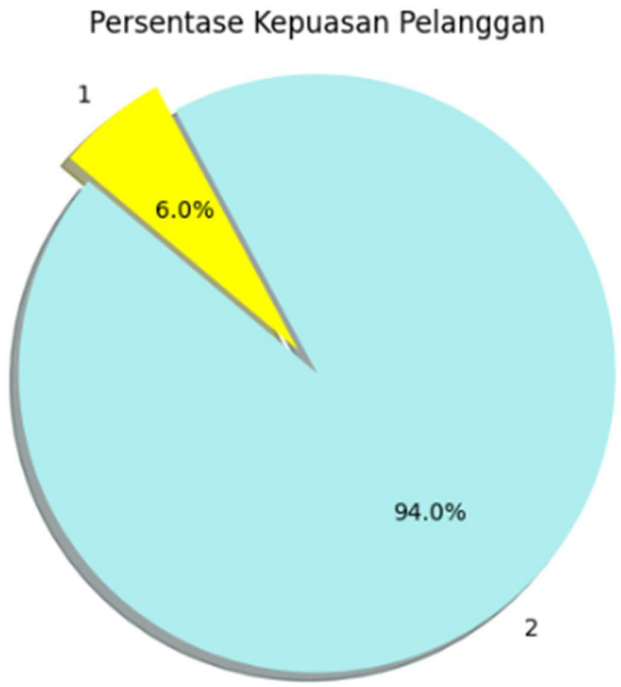
```
# Membuat diagram lingkaran untuk persentase kepuasan pelanggan
labels = kepuasan_percentage.index
sizes = kepuasan_percentage.values
colors = ['#AFEEEE','#FFFF00']
explode = (0.1, 0)  # Memisahkan potongan diagram untuk 'Puas'

plt.figure(figsize=(5, 5))
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140)
plt.title('Persentase Kepuasan Pelanggan')
plt.axis('equal')  # Menjaga agar pie chart berbentuk lingkaran
plt.show()
```

Gambar 3.15 Membuat Diagram Lingkaran Persentase Tingkat Kepuasan

Kode pada Gambar 3.17 digunakan untuk membuat diagram lingkaran (*pie chart*) yang menunjukkan persentase kepuasan pelanggan berdasarkan data yang telah dihitung sebelumnya. Diagram lingkaran ini membantu dalam visualisasi proporsi antara pelanggan yang merasa 'Puas' dan 'Tidak Puas'. Perintah ***labels*** digunakan untuk menyimpan label untuk setiap segmen dalam pie chart, yang diambil dari indeks **kepuasan_*percentage*** (yaitu, 'Puas' dan 'Tidak Puas'). ***sizes*** digunakan untuk menyimpan ukuran untuk setiap segmen dalam pie chart, yang diambil dari nilai **kepuasan_*percentage*** (yaitu, persentase dari masing-masing kategori). Perintah ***colors*** untuk menentukan warna yang akan digunakan untuk setiap segmen dalam *pie chart*. Di sini, '#AFEEEE' adalah warna untuk satu segmen dan '#FFFF00' untuk segmen lainnya. Perintah ***explode*** digunakan untuk memisahkan segmen tertentu dari *pie chart*

untuk penekanan. Dalam hal ini, segmen pertama ('Puas') dipisahkan sedikit (0.1) dari pusat, sedangkan segmen kedua ('Tidak Puas') tetap di tempatnya (0).

Perintah ***plt.figure(figsize*=(5, 5))** untuk menentukan ukuran figur *pie chart* yang akan dibuat, dengan ukuran 5x5 inci. Perintah ***plt.pie*** untuk membuat pie chart dengan parameter sebagai berikut: ***sizes***: ukuran setiap segmen, ***explode***: menentukan segmen yang akan dipisahkan, labels: label untuk setiap segmen, ***colors***: warna untuk setiap segmen, ***autopct='%1.1f%%'***: menampilkan persentase setiap segmen dengan format satu decimal, ***shadow=True***: menambahkan bayangan pada pie chart, ***startangle*=140**: menentukan sudut awal untuk pie chart, memutar chart untuk memulai dari sudut tertentu. Perintah ***plt.title*** untuk menetapkan judul untuk pie chart menjadi 'Persentase Kepuasan Pelanggan'. Perintah ***plt.axis('equal')*** untuk memastikan bahwa pie chart berbentuk lingkaran sempurna, bukan elips. Perintah ***plt.show*()** untuk menampilkan pie chart.



Gambar 3.16 Diagram Lingkaran Persentase Kepuasan Pelanggan

Gambar 3.18 merupakan diagram lingkaran yang menggambarkan hasil survei terkait tingkat kepuasan pelanggan. Dari hasil survei, ditemukan bahwa sebanyak 94% pelanggan merasa puas dengan produk/layanan yang diberikan dan hanya 6% pelanggan yang merasa tidak puas dengan produk/layanan tersebut. Hasil ini menunjukkan bahwa mayoritas besar pelanggan merasa puas, yang merupakan indikasi positif terhadap kualitas produk/layanan yang ditawarkan.

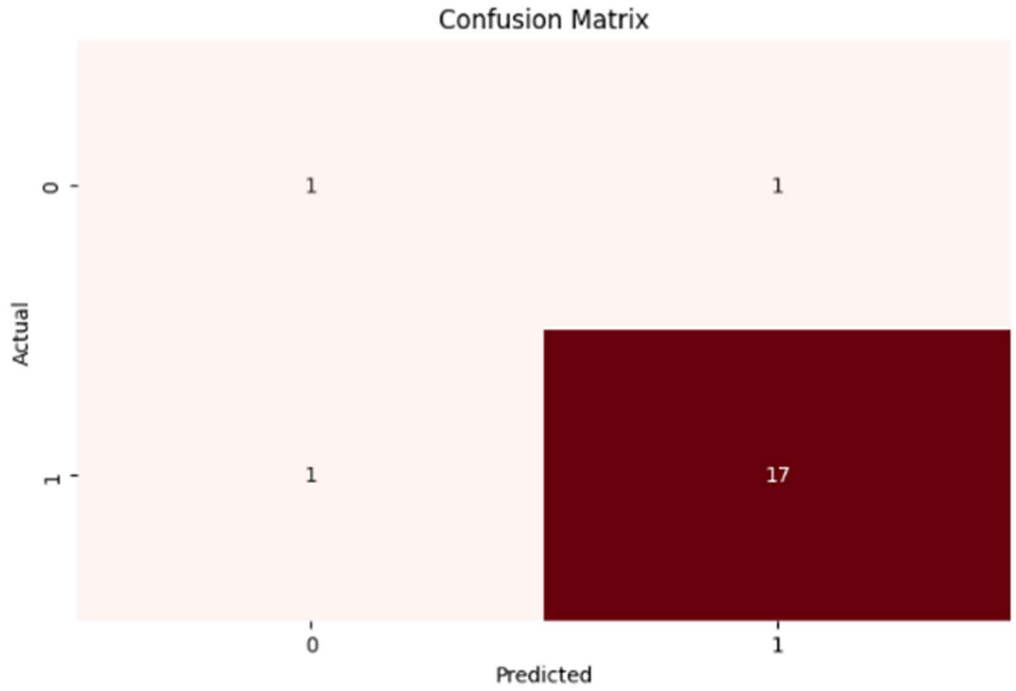
3.6.5 Membuat Diagram Confusion Matrix

```
# Membuat plot confusion matrix
plt.figure(figsize=(8, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Reds', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Gambar 3.17 Membuat Diagram *Confusion Matrix*

Kode pada Gambar 3.19 digunakan untuk membuat diagram *confusion matrix* menggunakan *seaborn heatmap*. *Confusion matrix* adalah alat visualisasi yang digunakan untuk mengevaluasi kinerja model klasifikasi dengan menampilkan jumlah prediksi yang benar dan salah untuk setiap kelas. Perintah ***plt.figure(figsize=(8, 5))*** digunakan untuk membuat figur baru dengan ukuran 8x5 inci untuk *heatmap*. Perintah ***sns.heatmap*** untuk membuat *heatmap* dari *confusion matrix conf_matrix*, perintah ***annot=True*** untuk menambahkan anotasi (angka) pada setiap sel dalam heatmap, perintah ***fmt='d'*** untuk menampilkan anotasi sebagai angka desimal (*integer*), perintah ***cmap='Reds'*** menentukan skema warna *heatmap* menggunakan warna merah, perintah ***cbar=False*** untuk menonaktifkan tampilan *color bar* di samping *heatmap*. *Color bar* biasanya digunakan untuk menunjukkan skala warna, tetapi dalam hal ini tidak diperlukan karena kita hanya fokus pada nilai angka. Perintah ***plt.xlabel('Predicted')*** untuk menetapkan label sumbu-x menjadi *'Predicted'* untuk menunjukkan bahwa sumbu-x mewakili kelas yang diprediksi oleh model. perintah ***plt.ylabel('Actual')*** untuk menetapkan

label sumbu-y menjadi *'Actual'* untuk menunjukkan bahwa sumbu-y mewakili kelas aktual yang benar. Perintah ***plt.xticks*(*ticks*=[0.5, 1.5], *labels*=['Tidak Puas', 'Puas'])** untuk menentukan posisi (*ticks*) dan label untuk sumbu-x, yaitu 'Tidak Puas' dan 'Puas'. Perintah ***plt.yticks*(*ticks*=[0.5, 1.5], *labels*=['Tidak Puas', 'Puas'])** untuk menentukan posisi (*ticks*) dan label untuk sumbu-y, yaitu 'Tidak Puas' dan 'Puas'. Perintah ***plt.title('Confusion Matrix')*** untuk menetapkan judul heatmap menjadi *'Confusion Matrix'*. Perintah ***plt.show()*** untuk menampilkan heatmap.



Gambar 3.18 *Confusion Matrix*

Gambar 3.20 merupakan *confusion Matrix* yang memberikan gambaran detail tentang kinerja model dengan menunjukkan jumlah prediksi benar dan salah untuk masing-masing kelas. *Confusion Matrix* menunjukkan bahwa model memiliki kelemahan dalam mengidentifikasi pelanggan yang puas dan tidak puas dengan tepat. Sebagian besar prediksi yang salah berasal dari pelanggan yang puas diprediksi sebagai tidak puas *(False Negatives)*

dan pelanggan yang tidak puas diprediksi sebagai puas *(False Positives)*. ***True Positives*** **(TP)**: 17. ***True Negatives*** **(TN)**: 1. ***False Positives*** **(FP)**: 1. ***False Negatives*** **(FN)**: 1.

3.6.6 Membuat Diagram Bar untuk Precision, Recall dan F1-Score

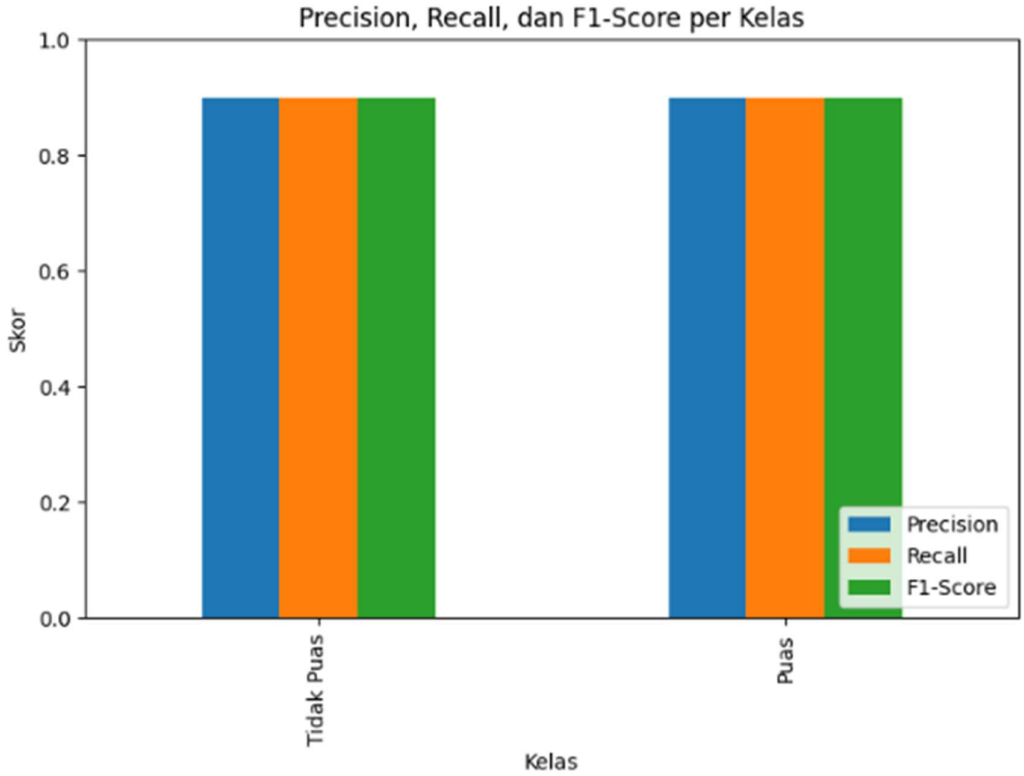
```
# Membuat diagram bar untuk Precision, Recall, dan F1-Score per kelas
metrics = pd.DataFrame({
    'Class': ['Tidak Puas', 'Puas'],
    'Precision': precision,
    'Recall': recall,
    'F1-Score': f1
})

metrics.set_index('Class', inplace=True)
metrics.plot(kind='bar', figsize=(8, 5))
plt.title('Precision, Recall, dan F1-Score per Kelas')
plt.xlabel('Kelas')
plt.ylabel('Skor')
plt.ylim(0, 1)
plt.legend(loc='lower right')
plt.show()
```

Gambar 3.19 Membuat Diagram *Bar* untuk *Precision, Recall* dan *F1-Score*

Kode pada Gambar 3.21 digunakan untuk membuat diagram batang (*bar chart*) yang menggambarkan metrik evaluasi *Precision, Recall, dan F1-Score* untuk setiap kelas dalam model klasifikasi. Perintah ***metrics = pd.DataFrame*** digunakan untuk membuat DataFrame dengan kolom ***Class, Precision, Recall, dan F1-Score***, ***Class*** berisi nama kelas ('Tidak Puas' dan 'Puas'), ***Precision, Recall, dan F1-Score*** berisi nilai metrik evaluasi untuk masing-masing kelas.

Perintah ***metrics.set_index('Class', inplace=True)*** untuk mengatur kolom ***Class*** sebagai indeks DataFrame untuk mempermudah *plot*. Perintah ***metrics.plot*(*kind='bar'*, *figsize*=(8, 5))** untuk membuat diagram batang dari DataFrame ***metrics*** dengan ukuran figur 8x5 inci. Perintah ***plt.title*** untuk menambahkan judul diagram **(*Precision, Recall, dan F1-Score* per Kelas)**,

perintah ***plt.xlabel*** untuk menambahkan label sumbu x (**Kelas**) dan perintah ***plt.ylabel*** untuk menambahkan label sumbu y (**Skor**). Perintah ***plt.ylim*** untuk menetapkan batas sumbu y dari 0 hingga 1 untuk menampilkan nilai metrik dengan lebih jelas. Perintah ***plt.legend(loc='lower right')*** untuk menambahkan legenda pada diagram di bagian kanan bawah. perintah ***plt.show*()** untuk menampilkan diagram batang yang telah dibuat.



Gambar 3.20 Diagram *Bar* untuk *Precision, Recall dan F1-Score*

Gambar 3.22 merupakan precission untuk mengukur proporsi prediksi positif yang benar dari semua prediksi positif yang dibuat oleh model. Pada diagram, terlihat bahwa skor presisi untuk kedua kelas "Tidak Puas" dan "Puas" mencapai 0.9, yang menunjukkan bahwa model memiliki tingkat kesalahan yang rendah dalam memprediksi kelas positif.. Adapun rumus *confusion matrix* yang digunakan adalah :

$$Precission = \frac{TP}{TP + FP} = \frac{17}{17 + 1} = \frac{17}{18} \approx 0.9$$

Recall mengukur proporsi positif yang benar-benar ditemukan oleh model dari semua kasus positif sebenarnya. Seperti halnya presisi, skor *recall* untuk kedua kelas juga mencapai 0.9, menunjukkan bahwa model mampu mengidentifikasi hampir semua kasus positif dengan benar. Adapun rumus *confusion matrix* yang digunakan adalah :

$$Recall = \frac{TP}{TP + FN} = \frac{17}{17 + 1} = \frac{17}{18} \approx 0.9$$

F1-Score adalah rata-rata harmonis dari presisi dan *recall*, memberikan gambaran keseimbangan antara keduanya. Dengan skor presisi dan *recall* yang tinggi, *F1-Score* juga mencapai 0.9 untuk kedua kelas, menunjukkan bahwa model bekerja sangat baik tanpa mengorbankan salah satu metrik. Adapun rumus *confusion matrix* yang digunakan adalah :

$$F1 - Score = 2 \times \left(\frac{Precission \times Recall}{Precission + Recall}\right) = 2 \times \left(\frac{0.944 \times 0.944}{0.944 + 0.944}\right) = 2 \times \left(\frac{0.891}{1.888}\right) = 2 \times 0.471 = 0.9$$

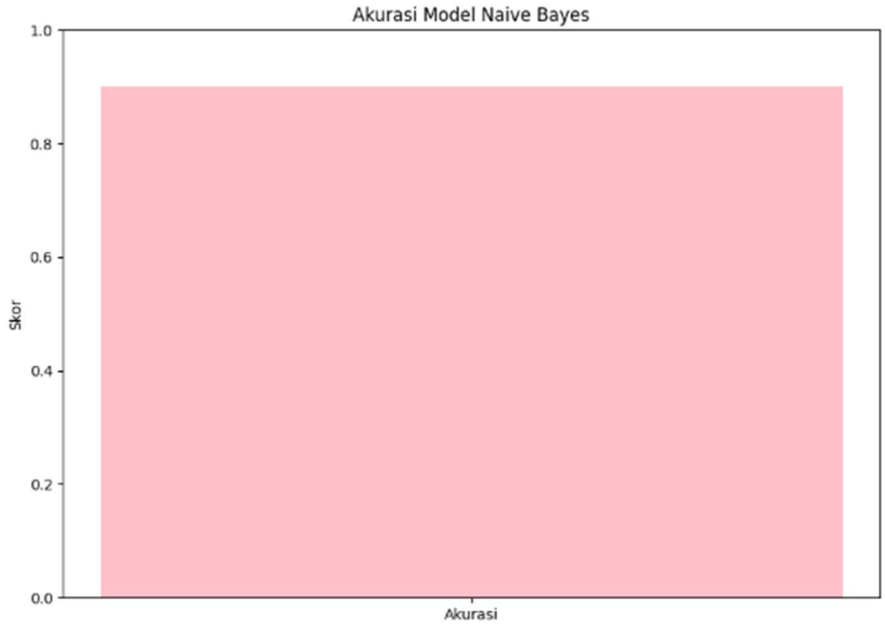
3.6.7 Membuat Diagram Akurasi

```
# Membuat diagram bar untuk Akurasi
plt.figure(figsize=(10, 7))
plt.bar(['Akurasi'], [accuracy], color='pink')
plt.ylim(0, 1)
plt.title('Akurasi Model Naive Bayes')
plt.ylabel('Skor')
plt.show()
```

Gambar 3.21 Membuat Diagram Akurasi

Kode pada Gambar 3.23 digunakan untuk membuat diagram batang (*bar chart*) yang menggambarkan akurasi model *Naive Bayes*. Diagram ini menunjukkan seberapa akurat model dalam memprediksi kelas berdasarkan data uji. Perintah ***plt.figure(figsize=(10, 7))*** digunakan untuk membuat figur diagram dengan ukuran 10x7 inci. Ini menetapkan ukuran kanvas di mana diagram batang akan digambar. Perintah ***plt.bar(['Akurasi'], [accuracy], color='pink')*** untuk membuat diagram batang dengan satu batang yang mewakili akurasi model, **['Akurasi']** untuk menetapkan label sumbu x sebagai "Akurasi", **[*accuracy*]** untuk menetapkan nilai sumbu y dengan nilai akurasi yang dihitung dari model, ***color='pink'*** untuk menetapkan warna batang

menjadi merah muda. Perintah ***plt.ylim*(0, 1)** untuk menetapkan batas sumbu y dari 0 hingga 1 untuk memastikan bahwa nilai akurasi ditampilkan dalam rentang standar (0-100%). Perintah ***plt.title*('Akurasi Model *Naive Bayes'*)** untuk menambahkan judul diagram untuk menjelaskan bahwa diagram ini menunjukkan akurasi model *Naive Bayes*, perintah ***plt.ylabel*('Skor')** untuk menambahkan label pada sumbu y sebagai "Skor". Perintah ***plt.show*()** untuk menampilkan diagram batang yang telah dibuat.



Gambar 3.22 Diagram Akurasi

Gambar 3.24 merupakan akurasi untuk mengukur proporsi prediksi yang benar (baik positif maupun negatif) dari semua prediksi yang dibuat oleh model. Dalam penelitian ini, model memiliki akurasi 0.9, yang berarti 90% dari semua prediksi adalah benar. Akurasi sebesar 90% menunjukkan bahwa model secara keseluruhan bekerja dengan baik. Adapun rumus confusion matrix yang digunakan adalah :

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{17 + 1}{17 + 1 + 1 + 1} = \frac{18}{17 + 1 + 1 + 1} = \frac{18}{20} \approx 0.9$$