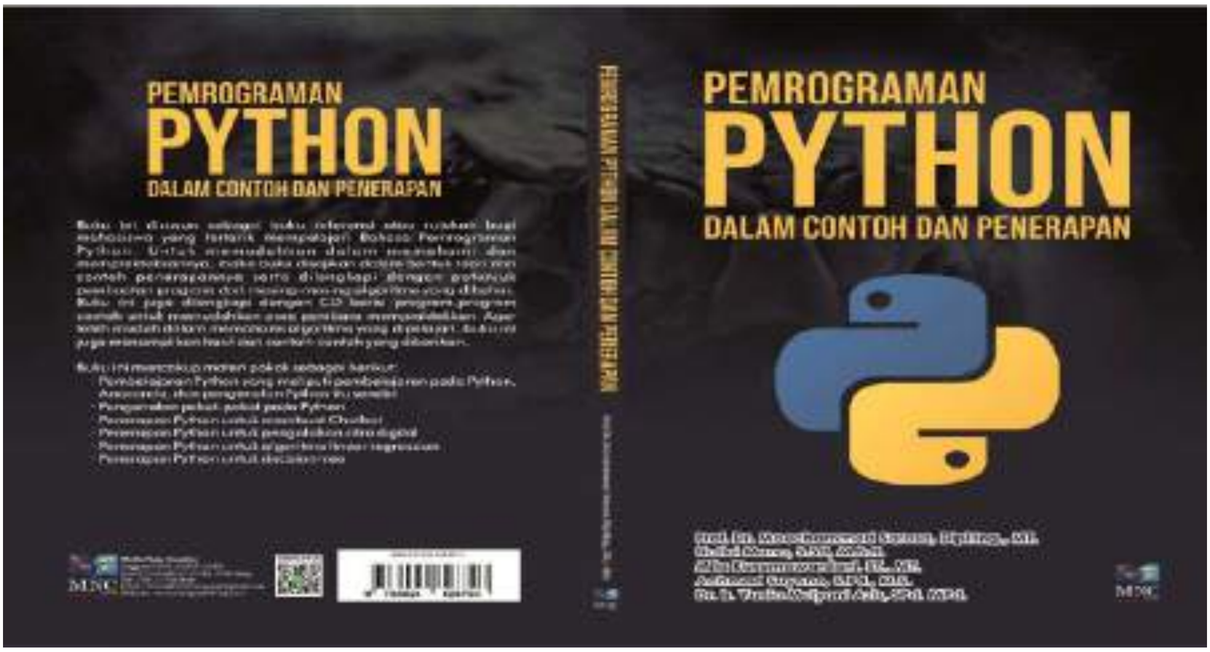
Table of Contents


PEMROGRAMAN
PYTHON
DALAM CONTOH DAN PENERAPAN
PEMROGRAMAN PYTHON DALAM CONTOH DAN PENERAPAN
PEMROGRAMAN
PYTHON
DALAM CONTOH DAN PENERAPAN

PEMROGRAMAN PYTHON DALAM CONTOH DAN PENERAPAN

Oleh

Prof. Dr. Moechammad Sarosa, Dipl.Ing., MT.
Nailul Muna, S.S.T, M.Tr.T.
Mila Kusumawardani, ST., MT.
Achmad Suyono, S.Pd., M.S.
Dr. Ir. Yunia Mulyani Azis, SPd. MPd.

MNC
PUBLISHING
FUTURE BOOKS WITH PASSION

PEMROGRAMAN PYTHON DALAM CONTOH DAN PENERAPAN

Penulis :
Prof. Dr. Moechammad Sarosa, Dipl.Ing., MT.
Nailul Muna, S.S.T, M.Tr.T.
Mila Kusumawardani, ST., MT.
Achmad Suyono, S.Pd., M.S.
Dr. Ir. Yunia Mulyani Azis, SPd. MPd.

Desain Cover & Penata Isi
Tim MNC Publishing

Cetakan I, Februari 2022

Diterbitkan oleh :

Media Nusa Creative
Anggota IKAPI (162/JTI/2015)
Bukit Cemara Tidar H5 No. 34, Malang
Telp. : 0812.3334.0088
E-mail : mncpublishing.layout@gmail.com
Website : www.mncpublishing.com

ISBN 978-602-462-870-3

KATA PENGANTAR

Puji syukur kita panjatkan kepada Allah SWT atas berkat dan rahmatNya kami berhasil menyelesaikan buku ini yang berjudul "**PEMROGRAMAN PYTHON DALAM CONTOH DAN PENERAPAN**".

Buku ini menyampaikan tentang dasar-dasar Bahasa pemrograman Python yang saat ini banyak diminati dan dibutuhkan oleh berbagai kalangan seperti mahasiswa, akademisi, peneliti, engineer dan software developer. Selain itu, pada buku ini juga memberikan contoh implementasi Python untuk membangun Chatbot.

Kami telah berupaya menyelesaikan buku ini dengan baik. Namun tidak menutup kemungkinan bahwa masih banyak kekurangan didalam buku ini. Oleh karena itu, kami menerima kritik dan saran yang mendukung teciptanya buku yang lebih baik lagi.

Harapannya semoga buku ini dapat bermafaat bagi pelajar, mahasiswa, maupun semua pihak yang ingin mempelajari Bahasa pemrograman Python dan implementasinya untuk membangun Chatbot.

Salam,

Penulis

DAFTAR ISI

BAB I

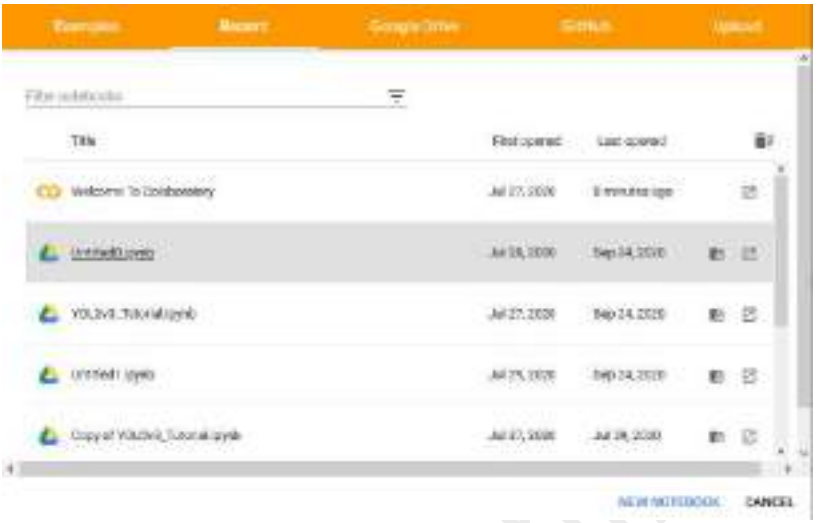
PEMBELAJARAN PYTHON

1.1 Google Colab

Google Colab merupakan environment dengan Bahasa pemrograman Python yang berbasis cloud serta dapat diakses secara gratis dan online. Berkas yang telah dikerjakan pada google Colab dapat diunggah pada Google Drive maupun Github. Penggunaan google colab dapat dilakukan sebagai berikut:

1. Membuka web browser dan melakukan pencarian dengan menuliskan Google Colab seperti yang ditampilkan pada Gambar 1.1. Kemudian buka Google Colab atau colab.research.google.com, maka akan muncul tampilan seperti pada Gambar 1.2.

Gambar 1.1 Pencarian Google Colab

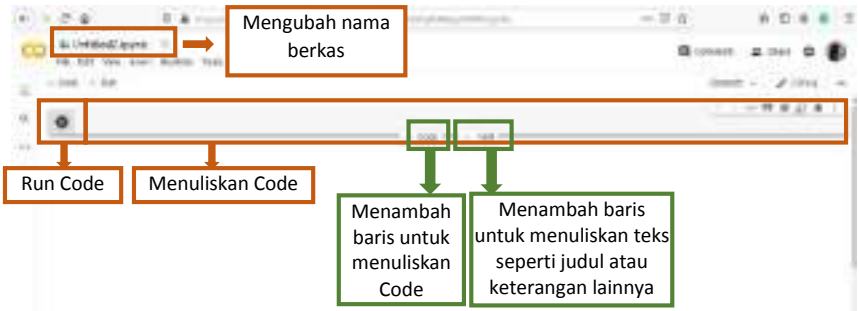


Gambar 1.2 Tampilan utama Google Colab

2. Membuat halaman notebook baru dengan pilih "NEW NOTEBOOK". Kemudian akan muncul lembar notebook Google Colabs seperti yang ditampilkan pada Gambar 1.3.

Gambar 1.3 Halaman Notebook Google Colab

3. Jika akan mengubah nama berkas sesuai kebutuhan, klik 2 kali pada untitled.ipynb kemudian nama berkas dapat diubah. Keterangan fungsi-fungsi pada Google Colab ditampilkan pada Gambar 1.4.



Gambar 1.4 Keterangan fungsi pada Google Colab

4. Berkas yang telah dikerjakan pada Google Colab tersimpan dengan ekstensi ".ipynb" atau interactive Python notebook.

1.2 Anaconda

Anaconda merupakan platform open source dengan Bahasa pemrograman Python. Anaconda dapat digunakan untuk machine learning dengan menggunakan Python.

Gambar 1.1 Ikon Anaconda

Beberapa paket yang digunakan untuk membangun dan melatih model machine learning diantaranya yaitu Spyder, Jupyter Notebook, JupyterLab, Numpy, TensorFlow, Pandas, Matplotlip, Conda. Seperti yang ditampilkan pada Gambar 1.6. Anaconda dapat didownload pada link berikut https://www.anaconda.com/products/individual.

Gambar 1.2 Paket pada Anaconda

1.3 Pengenalan Python

Beberapa hal yang perlu diketahui dari penulisan Bahasa Python yaitu sebagai berikut:

1.3.1 Penulisan Comment

Penulisan comment dapat dilakukan dengan menuliskan hashtag pada awal kalimat. Berikut contoh penulisan comment:

```
# Berikut adalah cara penulisan comment
```

Penulisan comment juga dapat dituliskan sebagai berikut:

```
'''
Berikut adalah paragraph pada python.
Penulisan ini juga dapat digunakan
sebagai Comment pada Python.
'''
```

1.3.2 Tipe Data Primitif

Pada Python terdapat beberapa tipe data primitif seperti halnya tipe data yang digunakan pada Bahasa pemrogaman yang lainnya. Berikut cara penulisan dari beberapa tipe data primitif menggunakan Bahasa pemrograman Python:

1.3.2.1 Integer

Integer merupakan tipe data bilangan bulat

```
variabel_1 = 500
```

1.3.2.2 Float

Float merupakan tipe data bilangan pecahan

```
variabel_2 = 15.75
```

1.3.2.3 String

String merupakan kumpulan dari karakter. String dituliskan di antara 2 tanda petik tunggal maupun ganda

```
variabel_3 = "Berikut adalah penulisan tipe data
string"
```

atau

```
variabel_4 = 'Berikut adalah penulisan tipe data
string'
```

1.3.2.4 Kompleks

Kompleks merupakan tipe data bilangan komplek yang terdiri dari bagian riil dan imajiner.

```
variabel_5 = 30 + 70j
```

1.3.2.5 Boolean

Boolean merupakan tipe data bilangan yang hanya memiliki 2 kemungkinan nilai yaitu true dan false

```
Boolean_1 = True
```

atau

```
Boolean_2 = False
```

1.3.2.6 Byte

```
byte_1 = a'abcdef12345'
```

Tipe data dapat diketahui dengan menuliskan type diikuti nama variabel pada tipe data yang ditanyakan. Berikut contoh penulisannya:

```
type(variabel_1)
```

atau

```
type(byte_1)
```

Untuk menampilkan nilai dari variabel yang berisi teks seperti tipe data **String** dan **Boolean** dapat dituliskan code sebagai berikut:

```
print(variabel_3)
```

atau

```
print(boolean_1)
```

1.3.3 Struktur Data

Beberapa struktur data pada Python diantaranya sebagai berikut:

1.3.3.1 List

List merupakan struktur data yang dapat berisi berbagai macam tipe data yang memiliki indeks. Cara penulisannya diawali dan diakhiri dengan kurung siku. Berikut cara penulisannya:

```
list_1 = [30, True, False, 20.5, 'Ini adalah
struktur data list', 17+5j]
```

```
list_2 = [3456, False, 12.5, 5+8j, 'Ini adalah
struktur data list', True]
```

Terdapat beberapa perintah pada list, diantaranya:

a) Concationation

Pada struktur data list terdapat operasi **concatination** yaitu menambahkan atau mengkombinasikan data pada list_1 dan list_2.

```
list_3 = list_1 + list_2
```

Untuk menampilkan hasil dari list_3 dapat menuliskan code print(list_3). Berikut hasil dari operasi concatination list_1 dan list_2.

```
[30, True, False, 20.5, 'Ini adalah struktur data
list', 17+5j, 3456, False, 12.5, 5+8j, 'Ini adalah
struktur data list', True]
```

b) Append

Append merupakan perintah untuk menambahkan data yang terletak diakhir deretan data (array) sebagai elemen tunggal. Berikut contoh perintah append:

```
[list_3.append(['Data Baru', 101010101])
print(list_3)
```

Berikut hasil dari perintah append diatas:

```
[30, True, False, 20.5, 'Ini adalah struktur data
list', 17+5j, 3456, False, 12.5, 5+8j, 'Ini adalah
struktur data list', True, ['Data Baru', 101010101]]
```

c) Extend

Extend merupakan perintah untuk memperpanjang data dengan menambahkan setiap elemen data ke dalam daftar.

```
list_3.extend(['Data Baru', 101010101])
list_3
```

Berikut hasil dari perintah extend diatas:

```
[30,
True,
False,
20.5,
'Ini adalah struktur data list',
17+5j,
3456,
False,
12.5,
5+8j,
'Ini adalah struktur data list',
True,
['Data Baru', 101010101],
'Data Baru',
101010101]
```

d) Len

Len merupakan perintah untuk menghitung data pada suatu deretan data (array). Berikut contoh perintah Len:

```
len (list_3)
```

Berikut hasil dari perintah len:

```
15
```

e) Insert

Insert merupakan perintah untuk menyisipkan data pada index posisi tertentu. Index pada deretan data dimulai dari index ke-0. Berikut contoh perintah insert untuk menyisipkan data pada index ke-2.

```
list_2.insert(2, 'Data X')
print(list_2)
```

Berikut hasil dari perintah diatas:

```
[3456, False, 'Data X', 12.5, 5+8j, 'Ini adalah
struktur data list', True]
```

f) Memperbarui Data

Memperbarui data dapat dilakukan menggunakan index posisi dari data yang akan diperbarui. Berikut contoh perintah untuk memperbarui data pada index ke-5.

```
list_2[4] = 'Data ke-5'
print(list_2)
```

Berikut hasil dari perintah diatas:

```
[3456, False, 'Data X', 12.5, 5+8j, 'Data ke-
5', True]
```

g) Hapus Data Berdasarkan Index Posisi Data

Menghapus data pada index posisi tertentu dan menampilkan data yang dihapus tersebut dapat dilakukan dengan menggunakan perintah pop. Berikut contoh perintah pop untuk menghapus data pada index ke-2.

```
list_2.pop(0)
```

Hasil dari perintah diatas yaitu sebagai berikut:

```
3456
```

Jadi deretan data list_2 yaitu sebagai berikut:

```
[False, 'Data X', 12.5, 5+8j, 'Data ke-5', True]
```

h) Hapus Data Berdasarkan Nilai Data

Hapus data juga dapat dilakukan berdasarkan nilai data menggunakan perintah remove. Berikut contoh perintah remove yang digunakan untuk hapus data pada list_2.

```
list_2.remove(12.5)
print(list_2)
```

Berikut hasil dari perintah diatas:

```
[False, 'Data X', 5+8j, 'Data ke-5', True]
```

Jika pada deretan data tidak ada data yang bernilai 0 atau 1, nilai 0 dan 1 dapat digunakan untuk menghapus data False dan True. 0 digunakan untuk menghapus data False dan 1 digunakan untuk menghapus data True.

i) Slicing List

Terdapat beberapa perintah pada slicing list, diantaranya sebagai be, diantaranya sebagai berikut:

» Untuk menampilkan data dalam 1 baris dapat dilakukan dengan perintah

```
print(list_3) atau print(list_3[:])
```

Berikut hasil dari perintah diatas:

```
[30, True, False, 20.5, 'Ini adalah struktur data list', 17+5j, 3456, False, 12.5, 5+8j, 'Ini adalah struktur data list', True, ['Data Baru', 101010101] , 'Data Baru', 101010101]
```

» Untuk menampilkan data dalam 1 kolom dapat dilakukan dengan perintah

```
list_3[:]
```

Berikut hasil dari perintah diatas:

```
[30,
True,
False,
20.5,
'Ini adalah struktur data list',
17+5j,
3456,
False,
12.5,
5+8j
'Ini adalah struktur data list',
True,
['Data Baru', 101010101],
'Data Baru',
101010101]
```

» Jika akan menampilkan beberapa data pada suatu deretan data dapat dilakukan dengan menuliskan perintah sebagai berikut.

```
List_3[0:6]
```

Perintah diatas digunakan untuk menampilkan data pada index posisi ke-0 sampai index posisi ke-6 dan berikut hasilnya:

```
[30, True, False, 20.5, 'Ini adalah struktur data list', 17+5j, 3456]
```

» Untuk menampilkan semua data kecuali data pada index terakhir dapat dilakukan dengan menuliskan perintah berikut:

```
List_3[:-1]
```

Berikut hasil dari perintah diatas:

```
[30,
True,
False,
20.5,
'Ini adalah struktur data list',
17+5j,
3456,
False,
12.5,
5+8j,
'Ini adalah struktur data list',
True,
['Data Baru', 101010101],
'Data Baru']
```

» Untuk menampilkan data dari index terbesar sampai index terkecil dapat dilakukan dengan perintah berikut:

```
list_3[-3:]
```

Berikut hasil dari perintah diatas:

```
[False, True, 30]
```

» Untuk menampilkan data dimulai data terakhir dapat menggunakan perintah reverse seperti contoh berikut:

```
list_3.reverse()
print(list_3)
```

Berikut hasil dari perintah reverse:

```
[30, True, False, 20.5, 'Ini adalah struktur data list', 17+5j, 3456, False, 12.5, 5+8j, 'Ini adalah struktur data list', True, ['Data Baru', 101010101], 'Data Baru', 101010101]
```

» Untuk menyalin data dapat dilakukan menggunakan perintah copy seperti contoh berikut:

```
list_4 = list_3.copy()
```

» Untuk menghapus data dapat dilakukan dengan perintah clear seperi contoh berikut:

```
list_4.clear()
print (list_4)
```

Berikut hasil dari perintah clear

```
[]
```

» Untuk menghapus variabel secara permanen dapat dilakukan dengan menggunakan perintah del seperti contoh berikut:

```
del(list_4)
```

1.3.3.2 Tuple

Tuple merupakan struktur data yang tidak dapat diubah. penulisan struktur data tuple yaitu sebagai berikut:

```
tuple_1=(987, 'List Data','new value',True, False
,233455,'new list',121345)
```

Perintah-perintah pada tuple yaitu sebagai berikut:

a) Menghitung Jumlah Data

Menghitung jumlah data tertentu pada deretan data pada suatu variabel dapat mrnggunakan perintah count. Misalkan menghitung data True pada variabel tuple_1 dapat dituliskan perintah seperti berikut:

```
tuple_1.count(True)
```

Hasil perintah count yaitu sebagai berukut:

```
1
```

b) Mengetahui Index Posisi Data

Mengetahui index posisi data pada deretan data dapat menggunakan perintah index seperti contoh berikut:

```
tuple_1. index (False)
```

Hasil perintah index yaitu sebagai berukut:

```
4
```

1.3.3.3 Dictionary

Dictionary merupakan struktur data yang berisi kunci (key) dan nilai (value). Berikut contoh struktur data dictionary.

```
dict_1={'A': 'Apel', 'B':'Buah','C':'Ceri', 'D' :
'Durian'}
```

```
dict_2=dict([(1, 'Cina'),(2,'Indonesia'),(3,'Rusia')])
```

Perintah-perintah pada strukutur data dictionary yaitu sebagai berikut:

a) Key

Untuk menampilkan kunci (key) dari dictionary dapat menggunakan perintah keys seperti contoh berikut:

```
dict_2.keys()
```

Berikut hasil dari perintah keys:

```
dict_keys([1, 2, 3])
```

b) Value

Untuk menampilkan nilai (value) dari dictionary dapat menggunakan perintah values seperti contoh berikut:

```
dict_2. values ()
```

Berikut hasil dari perintah values:

```
dict_values(['Cina', 'Indonesia', 'Rusia'])
```

c) Update

Update digunakan untuk menambah atau mengubah data pada deretan data dictionary. Berikut penulisan struktur data update.

```
dict_2.update({4:'Singapura'})
dict_2
```

Berikut hasil dari perintah values:

```
{1: 'China', 2: 'Indonesia, 3: 'Rusia', 4: Singapura}
```

1.3.4 Loop

Loop pada Python melakukan iterasi hingga kondisi menjadi False. Berikut contoh loop menggunakan perintah for.

```
#int i = 0, i <20, i = i+1
for i in range(0, 20, 1):
  print(i)
```

Keterangan:
0 merupakan kondisi awal
20 merupakan kondisi akhir
1 merupakan step penambahan nilai
Hasil looping dari perintah diatas yaitu sebagai berikut:

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

1.3.5 Ganjil dan Genap

Untuk mengetahui bilangan ganjil (odd) dan genap (even) pada Python bisa menggunakan perintah modulus (%) seperti contoh berikut:

```
num = 20
  if num % 2 == 0:
      print("Bilangan Genap: %i" %num)
```

Perintah diatas digunakan untuk mengetahui bilangan yang diinputkan merupakan bilangan genap atau ganjil. Perintah yang diguanakan yaitu num % 2 == 0 yang berarti variabel yang

diinputkan akan habis atau sisa sama dengan 0 jika dibagi 2. Hasil dari perintah diatas yaitu sebagai berikut:

```
Bilangan Genap: 20
```

Perintah for juga dapat digunakan pada perintah modulus. Berikut contoh perintah modulus menggunakan perintah for untuk looping.

```
for num in range(1, 20, 1):
  if num % 2 == 0:
    print("Bilangan Genap: %i" %num)
  else:
    print("Bilangan Ganjil: %i" %num)
```

Pada perintah diatas, jika angka yang diproses habis dibagi 2 merupakan bilangan genap, jika tidak atau angka tersebut tidak sama dengan 0, maka angka tersebut merupakan bilangan ganjil. Hasil dari perintah diatas yaitu sebagai berikut

```
Bilangan Ganjil: 1
Bilangan Genap: 2
Bilangan Ganjil: 3
Bilangan Genap: 4
Bilangan Ganjil: 5
Bilangan Genap: 6
Bilangan Ganjil: 7
Bilangan Genap: 8
Bilangan Ganjil: 9
Bilangan Genap: 10
Bilangan Ganjil: 11
Bilangan Genap: 12
Bilangan Ganjil: 13
Bilangan Genap: 14
Bilangan Ganjil: 15
Bilangan Genap: 16
Bilangan Ganjil: 17
Bilangan Genap: 18
Bilangan Ganjil: 19
```

1.3.6 def Keywoard

Deklarasi fungsi pada Python dapat menggunakan perintah def. Berikut contoh deklarasi dari perintah def yang digunakan untuk membuat salam.

```
def salam (nama):
  print("Halo " + str(nama) + " Selamat Pagi")
```

Keterangan:
salam merupakan nama fungsi
nama atau dapat disebut dengan parameter yaitu variabel yang menjadi input yang akan diproses didalam fungsi.

Untuk memanggil fungsi dapat dilakukan dengan menuliskan nama fungsi yang diikuti dengan variabel yang akan diiputkan seperti contoh berikut:

```
salam("Eka.")
```

Hasil dari perintah diatas yaitu sebagai berikut:

```
Halo Eka. Selamat Pagi
```

BAB II

PAKET PADA PYHTON

Python memiliki paket atau library untuk import dan eksport suatu data seperti Pandas, NumPy, dan SciPy.

2.1 Pandas

Pandas merupakan suatu paket yang digunakan untuk menganalisis dan memanipulasi data yang dibangun diatas Bahasa pemrograman Python. Untuk mengimport paket atau library panda dapat dilakukan dengan perintah berikut:

```
import pandas as pd
```

Misalkan diiputkan data sebagai berikut:

```
data = [10, 20, 30, 40, 50, 60]
```

2.1.1 Representasi Data

Pengumpulan data 1-D atau skalar satu kolom menggunakan Pandas dari data data yang telah diinputkan sebelumnya yaitu sebagai berikut.

```
pd.Series(data)
```

dan hasilnya sebagai berikut:

```
0    10
1    20
2    30
3    40
4    50
5    60
dtype: int64
```

Raw index data 1-D dapat diubah menggunakan perintah sebagai berikut:

```
pd.Series(data = [10, 20, 30, 40, 50], index = ['
a','b','c','d','e'], dtype = 'int16')
```

dan hasilnya sebagai berikut:

```
a    100
b    200
c    300
d    400
e    500
```

Representasi menggunakan 2 kolom dapat menggunakan perintah sebagai berikut:

```
data_series = {'Column1': pd.Series(data = [1000, 20
00, 3000, 4000, 5000], index = ['a','b','c','d','e']
, dtype = 'int16'),
               'Column2': pd.Series(data = [1, 2, 3,
 4, 5], index = ['a','b','c','d','e'], dtype = 'int1
6')}

print(data_series)
```

Hasilnya yaitu sebagai berikut:

```
{'Column1':
a    1000
b    2000
c    3000
d    4000
e    5000
dtype: int16,
'Column2': a    1
b    2
c    3
d    4
e    5
dtype: int16}
```

Data Frame adalah representasi dari data N-D seperti Array, Series, dll. menjadi representasi yang tersusun secara berurut berdasar kolom dan baris. Representasi menggunakan data frame dapat menggunakan perintah sebagai berikut.

```
pd.DataFrame(data_series)
```

Hasilnya yaitu sebagai berikut:

	Column1	Column2
a	1000	1
b	2000	2
c	3000	3
d	4000	4
e	5000	5

Gambar 2.1 Representasi data menggunakan data frame

Nama kolom pada data frame dapat diubah dengan perintah sebagai berikut:

```
Penjualan = {'Id_Pelanggan' : [111, 112,113],
                'Id_Produk': ['x01', 'x02', 'x03']}

pd.DataFrame(Penjualan)
```

Hasilnya yaitu sebagai berikut:

	Id_Pelanggan	Id_Produk
0	111	x01
1	112	x02
2	113	x03

Gambar 2.2 Representasi data menggunakan data frame dengan mengubah nama kolom

2.1.2 Import Berkas CSV

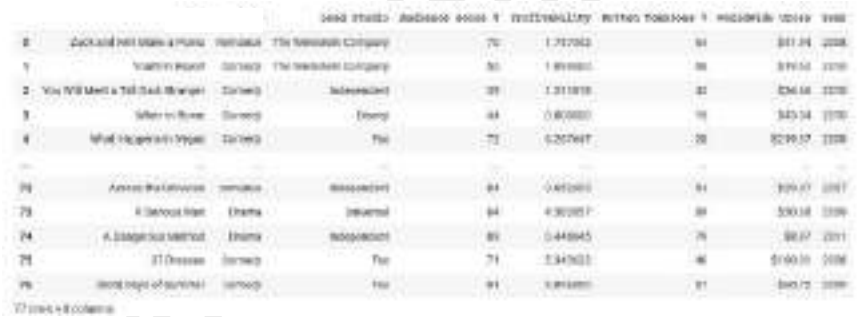
Import berkas dapat dilakukan dari beberapa sumber seperti csv, txt, excel, access, dan lain-lain. Berikut contoh import data menggunakan csv.

```
movies_df = pd.read_csv('https://raw.githubusercontent.com/ammishra08/MachineLearning/master/Datasets/movies.csv')
```

Perintah diatas digunakan untuk membaca berkas movies.csv pada github. Untuk menampilkan data dapat dilakukan dengan menuliskan perintah berikut:

```
movies_df
```

Dan berikut data dari berkas movies.csv.



Gambar 2.3 Data dari berkas movies.csv

2.1.2.1 Fungsi Head ()

Pada data frame terdapat fungsi head() yang digunakan untuk menampilkan beberapa index awal. Tanda kurung pada fungsi head() menampilkan data beberapa baris dari awal. Jika pada tanda kurung diisi 7, maka data frame akan menampilkan 7 index pertama. Jika pada tanda kurung tidak mengatur jumlah index yang akan ditampilkan, maka data frame akan menampilkan jumlah index secara default yaitu 5 baris. Berikut contoh dari fungsi head().

```
movies_df.head()
```

Berikut tampilan data frame menggunakan fungsi head().

Gambar 2.4 Tampilan data frame menggunakan fungsi head()

2.1.2.2 Fungsi Tail ()

Pada data frame terdapat fungsi tail() yang digunakan untuk menampilkan beberapa index terakhir. Seperti fungsi head(), tanda kurung pada fungsi tail() dapat menampilkan data beberapa baris dari akhir. Jika pada tanda kurung diisi 10, maka data frame akan menampilkan 10 index terakhir. Jika pada tanda kurung tidak mengatur jumlah index yang akan ditampilkan, maka jumlah index yang ditampilkan yaitu default berjumlah 5 baris. Berikut contoh dari fungsi tail().

```
movies_df. tail ()
```

Berikut tampilan data frame menggunakan fungsi tail().

	film	genre	lead studio	audience score %	profitability	rotten tomatoes %	worldwide gross	year
72	Across the Universe	romance	Independent	84	0.493600	34	$29.37	2007
73	A Serious Man	Drama	Universal	64	4.962857	89	$30.68	2009
74	A Dangerous Method	Drama	Independent	68	0.448645	79	$31.97	2011
75	21 Grams	Comedy	Fox	71	5.319622	82	$159.11	2009
76	(500) Days of Summer	comedy	Fox	81	8.098000	87	$60.72	2009

Gambar 2.5 Tampilan data frame menggunakan fungsi tail()

2.1.3 Manipulasi Data

Untuk mengetahui jumlah baris dan kolom pada suatu data frame dapat menggunakan perintah shape seperti berikut.

```
movies_df.shape
```

Maka akan tampil jumlah baris dan kolom dari data frame yang digunakan seperti berikut:

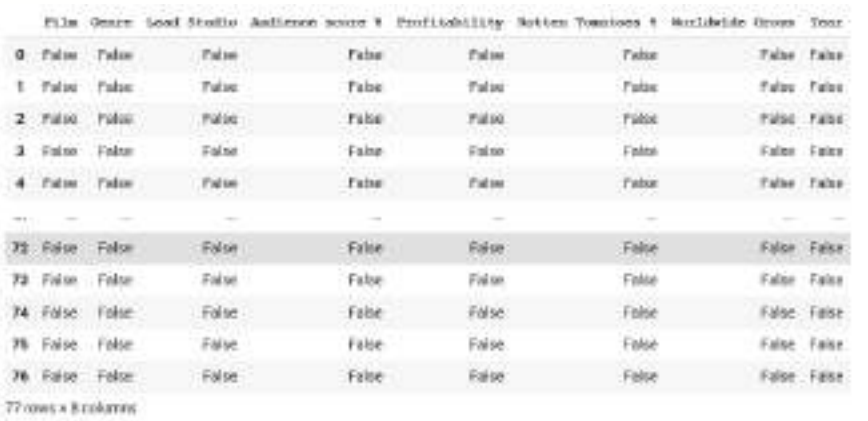
```
(77, 8)
```

2.1.3.1 Fungsi isnull ()

Fungsi isnull()digunakan untuk mendeteksi nilai yang hilang berdasarkan kolom. Berikut contoh penggunaan fungsi isnull().

```
movies_df.isnull()
```

Berikut tampilan data frame dari fungsi isnull().



	Film	Genre	Lead Studio	Audience score %	Profitability	Rotten Tomatoes %	Worldwide Gross	Year
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
72	False	False	False	False	False	False	False	False
73	False	False	False	False	False	False	False	False
74	False	False	False	False	False	False	False	False
75	False	False	False	False	False	False	False	False
76	False	False	False	False	False	False	False	False

77 rows x 8 columns

Gambar 2.6 Tampilan data frame menggunakan fungsi isnull()

Berdasarkan Gambar 2.6, dapat diketahui data frame semua variabel bernilai false yang berarti tidak ada variabel yang kosong atau tidak bernilai. Dari data tersebut dapat diakumulasi menggunakan fungsi berikut.

```
movies_df.isnull().sum()
```

Berikut hasil dari fungsi diatas.

```
Film                0
Genre               0
Lead Studio         0
Audience score %    0
Profitability       0
Rotten Tomatoes %   0
Worldwide Gross     0
Year                0
dtype: int64
```

Berdasarkan hasil diatas dapat diketahui bahwa dari semua variabel memiliki nilai 0 yang berarti tidak ada variabel yang tidak memiliki data atau nilai.

2.1.4 Memilih Kolom

Pada data frame data memilih kolom mana yang akan ditampilkan. Pemilihan kolom tersebut dapat dilakukan menggunakan perintah sebagai berikut.

```
X = movies_df[['Film', 'Genre']]

display(X)
```

Berdasarkan perintah diatas, kolom yang dipilih untuk ditampilkan yaitu kolom 'Film' dan 'Genre' seperti data frame berikut.

	Film	Genre
0	Zack and Miri Make a Porno	Ron
1	Youth in Revolt	Comedy
2	You Will Meet a Tall Dark Stranger	Comedy
3	When in Rome	Comedy
4	What Happens in Vegas	Comedy
...	...	...
72	Across the Universe	romance
73	A Serious Man	Drama
74	A Dangerous Method	Drama
75	27 Dresses	Comedy
76	(500) Days of Summer	comedy

77 rows × 2 columns

Gambar 2.7 Hasil pemilihan kolom 'Film' dan 'Genre'

2.1.4.1 Fungsi unique ()

Fungsi unique() digunakan untuk menampilkan nilai pada suatu kolom yang dipilih. Berikut contoh penggunaan fungsi unique().

```
movies_df['Genre'].unique()
```

Berdasarkan perintah diatas, fungsi unique()digunakan untuk menampilkan nilai dari kolom 'Genre'. Berikut hasil dari perintah diatas.

```
array(['Romance', 'Comedy', 'Drama', 'Animation',
'Fantasy', 'Romence', 'Comdy', 'Action', 'romance',
'comedy'], dtype=object)
```

2.1.4.2 Filter

Filter digunakan untuk menampilkan data sesuai yang dibutuhkan. Filter dapat dilakukan berdasarkan inisial nama kolom. Berikut contoh fungsi filter menggunakan inisial nama kolom.

```
movies_df.filter(like='F')
```

Berikut hasil fungsi filter.

	Film
0	Zack and Miri Make a Porno
1	Youth in Revolt
2	You Will Meet a Tall Dark Stranger
3	When in Rome
4	What Happens in Vegas
...	...
72	Across the Universe
73	A Serious Man
74	A Dangerous Method
75	27 Dresses
76	(500) Days of Summer

77 rows × 1 columns

Gambar 2.8 Hasil filter nama kolom berdasarkan inisial

Filter juga dapat dilakukan berdasarkan nama kolom dan diurutkan berdasarkan kecil ke besar atau sebaliknya. Filter kolom dan pengurutan nilai dapat dilakukan dengan perintah berikut.

```
movies_df.filter(like = 'Genre').sort_values(by =
'Genre', ascending = False)
```

Berdasarkan perintah diatas kolom yang ditampilkan yaitu kolom 'Genre' dan pengurutan nilainya yaitu ascending = False yang berarti nilai atau data pada kolom 'Genre' diurutkan dari besar ke kecil. Hasil dari perintah diatas yaitu sebagai berikut.

	Genre
72	romance
76	comedy
43	Romence
11	Romance
57	Romance
...	...
65	Animation
6	Animation
22	Animation
64	Animation
55	Action

77 rows × 1 columns

Gambar 2.9 Hasil filter nama kolom berdasarkan inisial dan pengurutan data

2.1.5 Slicing Data

Berikut beberapa fungsi yang digunakan untuk slicing data.

2.1.5.1 loc untuk Memilih Baris dan Kolom pada Pandas

loc pada Pandas digunakan untuk memilih data berdasarkan label pada index data dan/atau berdasarkan Boolean atau nama kolom pada data frame. Untuk menampilkan semua data pada data frame dapat dilakukan dengan menggunakan perintah sebagai berikut.

```
movies_df.loc[: , :]
```

Hasil dari perintah diatas yaitu sebagai berikut.

Gambar 3.10 Hasil perintah loc

Untuk menampilkan index tertentu dan kolom apa saja yang dibutuhkan dapat menggunakan perintah berikut:

```
movies_df.loc [1:10,['Film','Audience score %', '
Profitability']
```

Berdasarkan perintah diatas, baris yang akan ditampilkan yaitu pada baris 1 sampai 10 dan kolom yang akan ditampilkan yaitu kolom 'Film','Audience score %', dan 'Profitability'. Tampilan dari perintah diatas yaitu sebagai berikut.

	Film	Audience score %	Profitability
1	Youth in Revolt	52	1.090000
2	You Will Meet a Tall Dark Stranger	35	1.211818
3	When in Rome	44	0.000000
4	What Happens In Vegas	72	6.267647
5	Water For Elephants	72	3.081421
6	WALL-E	89	2.896019
7	Waitress	67	11.089741
8	Waiting For Forever	53	0.005000
9	Valentine's Day	54	4.184038
10	Tyler Perry's Why Did I get Married	47	3.724192

Gambar 2.11 Hasil perintah loc untuk menampilkan kolom dan baris yang berurutan

Pemilihan baris dan kolom tertentu juga dapat dilakukan menggunakan perintah berikut.

```
movies_df.loc [[2,7,12],['Film','Audience score %'
, 'Profitability']]
```

Pada perintah diatas hanya digunakan untuk menampilkan index 2, 7, dan 12. Berikut hasilnya.

	Film	Audience score %	Profitability
2	You Will Meet a Tall Dark Stranger	35	1.211818
7	Waitress	67	11.089741
12	Twilight	82	10.180027

Gambar 2.12 Hasil perintah loc untuk menampilkan kolom dan baris yang tidak berurutan

2.1.5.2 iloc untuk Memilih Baris dan Kolom pada Pandas

iloc pada Pandas digunakan untuk memilih baris dan kolom berdasarkan posisi tertentu pada indeks data frame. Berikut contoh perintah iloc.

```
movies_df.iloc[1:10, 0:5]
```

Berdasarkan perintah diatas data yang ditampilkan yaitu data pada posisi indeks ke-1 sampai ke-10 dan kolom pada posisi indeks ke-0 sampai ke-5. Berikut hasilnya:

	Film	Genre	Lead Studio	Audience score %	Profitability
1	Youth in Revolt	Comedy	The Weinstein Company	52	1.090000
2	You Will Meet a Tall Dark Stranger	Comedy	Independent	35	1.211818
3	When in Rome	Comedy	Disney	44	0.000000
4	What Happens in Vegas	Comedy	Fox	72	6.267647
5	Water For Elephants	Drama	20th Century Fox	72	3.081421
6	WALL-E	Animation	Disney	89	2.896019
7	Waitress	Romance	Independent	67	11.089741
8	Waiting For Forever	Romance	Independent	53	0.005000
9	Valentine's Day	Comedy	Warner Bros.	54	4.184038

Gambar 2.13 Hasil perintah iloc untuk menampilkan kolom dan baris tertentu

Untuk menampilkan semua baris data dan beberapa kolom terakhir dari data frame dapat menggunakan perintah sebagai berikut.

```
movies_df.iloc[:,-4:]
```

Pada perintah diatas, data yang ditampilkan yaitu keseluruhan baris dan 4 kolom terakhir.

	Profitability	Rotten Tomatoes %	Worldwide Gross	Year
0	1.747542	64	$41.94	2008
1	1.090000	68	$19.62	2010
2	1.211818	43	$26.66	2010
3	0.000000	15	$43.04	2010
4	6.267647	28	$219.37	2008
...	...	...	...	...
72	0.652603	54	$29.37	2007
73	4.382857	89	$30.68	2009
74	0.448645	79	$8.97	2011
75	5.343622	40	$160.31	2008
76	8.096000	87	$60.72	2009

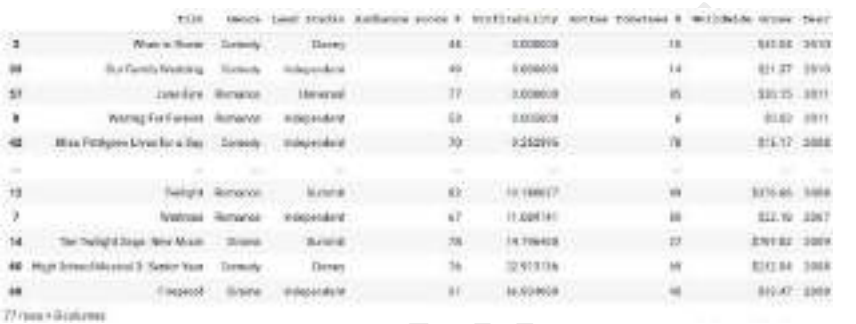
77 rows × 4 columns

Gambar 2.14 Hasil perintah iloc untuk menampilkan keseluruhan baris dan kolom tertentu

Untuk menyortir data berdasarkan parameter tertentu dapat dilakukan menggunakan perintah berikut.

```
movies_df.sort_values(by = 'Profitability', ascendi
ng = True)
```

Parameter diatas digunakan untuk menyortir data berdasarkan parameter profitability secara ascending.



Gambar 2.15 Hasil perintah sortir nilai parameter profitability secara ascending

2.1.6 Fungsi Drop

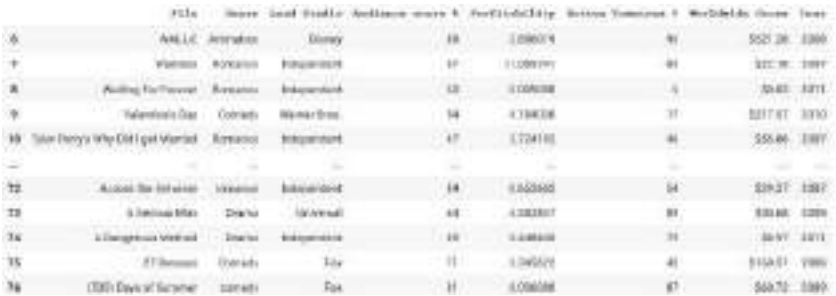
Fungsi drop Pandas digunakan untuk menyembunyikan data pada indeks tertentu. Fungsi drop dapat diterapkan pada baris maupun kolom data frame.

2.1.6.1 Drop Baris

drop baris dapat dilakukan menggunakan perintah berikut.

```
movies_df.drop([0,1,2,3,4,5],axis=0)
```

Berdasarkan perintah diatas, data frame tidak menampilkan index data 0,1,2,3,4, dan 5. Sehinga berikut tampilan data frame setelah melalui fungsi drop.



Gambar 2.16 Hasil fungsi drop baris

2.1.6.2 Drop Kolom

Drop kolom dapat dilakukan menggunakan perintah berikut.

```
movies_df.drop(['Lead Studio','Rotten Tomatoes %'],
axis=1)
```

Berdasarkan perintah diatas, data frame tidak menampilkan kolom 'Lead Studio','Rotten Tomatoes %'. Sehinga berikut tampilan data frame setelah melewati fungsi drop kolom.

	Film	Genre	Audience score %	Profitability	Worldwide Gross	Year
0	Zach and Miri Make a Porno	Romance	70	1.747540	$41.94	2008
1	Youth in Revolt	comedy	52	1.090000	$19.62	2010
2	You Will Meet a Tall Dark Stranger	Comedy	35	1.211818	$26.66	2010
3	When in Rome	Comedy	44	0.000000	$43.04	2010
4	What Happens in Vegas	Comedy	72	6.267647	$219.37	2008
...	...	...	...	...	...	...
72	Across the Universe	romance	84	0.858000	$29.37	2007
73	A Serious Man	Drama	64	4.292857	$30.68	2009
74	A Dangerous Method	Drama	89	0.448649	$8.97	2011
75	27 Dresses	Comedy	71	5.343622	$160.31	2008
76	(500) Days of Summer	comedy	81	8.096000	$60.72	2009

77 rows × 6 columns

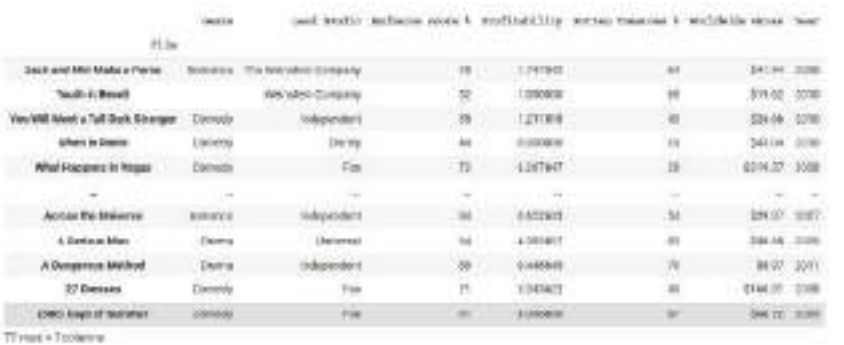
Gambar 2.17 Hasil fungsi drop kolom

2.1.7 Fungsi set_index

Fungsi set_index digunakan untuk mengatur indeks berdasarkan parameter yang dipilih.

```
movies_new = movies_df.set_index('Film')
display(movies_new)
```

Perintah diatas menggunakan parameter 'Film' sebagai indeksnya. Sehingga tampilan dari data framenya yaitu sebagai berikut.



Gambar 2.18 Hasil set_index 'Film'

Fungsi loc dapat digunakan untuk menampilkan informasi pada index tertentu. Berdasarkan Gambar 2.18, index telah diubah menggunakan nama film. Sehingga dengan menggunakan loc, dapat mengetahui informasi sesuai kolom yang tersedia mengenai film yang dipilih tersebut. Berikut contohnya.

```
movies_new.loc['27 Dresses']
```

Berikut informasi yang ditampilkan.

```
Genre                    Comedy
Lead Studio                 Fox
Audience score %             71
Profitability           5.34362
Rotten Tomatoes %            40
Worldwide Gross         $160.31
Year                       2008
Name: 27 Dresses, dtype: object
```

Berbeda dengan loc, fungsi iloc digunakan untuk menampilkan informasi berdasarkan index penomoran. Berikut contohnya.

```
movies_new.iloc[5]
```

Berikut informasi yang ditampilkan

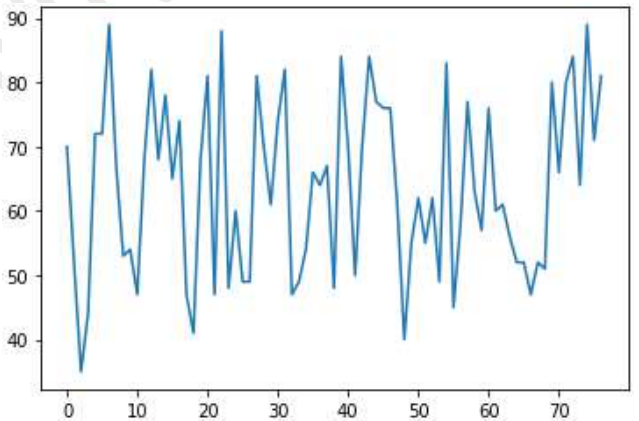
```
Genre                              Drama
Lead Studio              20th Century Fox
Audience score %                      72
Profitability                    3.08142
Rotten Tomatoes %                     60
Worldwide Gross                  $117.09
Year                                2011
Name: Water For Elephants, dtype: object
```

2.1.8 Visualisasi Data Menggunakan Pandas

Visualisasi data menggunakan Pandas dapat dilakukan menggunakan perintah berikut.

```
movies_df['Audience score %'].plot()
```

Berdasarkan perintah diatas visualisasi data yang akan ditampilkan yaitu data Audience score %. Secara default visualisasi data yang ditampilkan yaitu grafik garis. Berikut hasilnya.

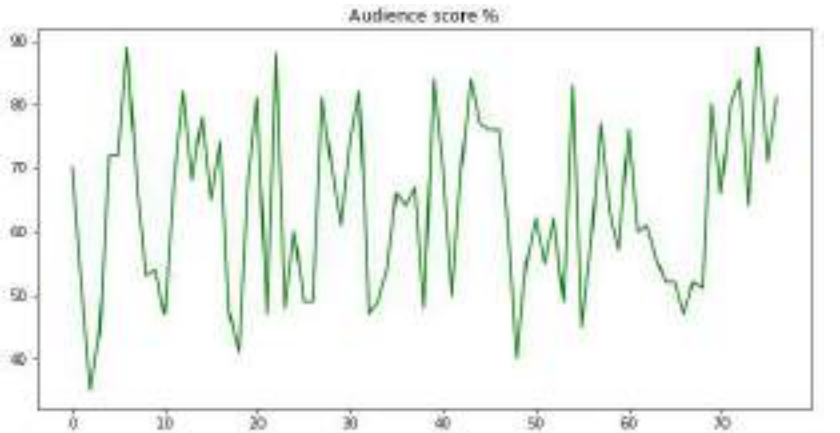


Gambar 2.19 Visualisasi data secara default

Dari grafik pada Gambar 2.19, ukuran grafik, judul grafik, dan warna grafik dapat diatur sesuai dengan kebutuhan dengan perintah berikut.

```
movies_df['Audience score %'].plot(figsize=(10,5),
  title = 'Audience score %', color = 'Green')
```

Berikut tampilannya.

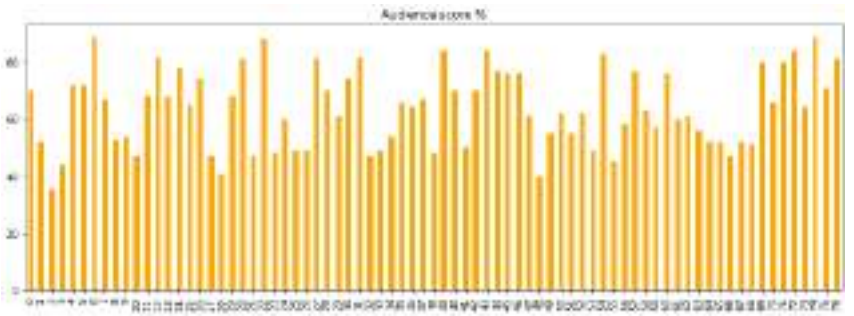


Gambar 2.20 Grafik garis setelah ukuran, judul, dan warna diatur

Visualisasi data juga data juga dapat ditampilkan menggunakan grafik batang. Berikut contoh peritah membuat grafik batang.

```
movies_df['Audience score %'].plot(kind='bar',fig
size=(15,5), title = 'Audience score %', color =
'orange' )
```

Pada perintah diatas data yang akan ditampilkan yaitu Audience score % dalam bentuk bar atau grafik batang. Ukuran grafik sebesar (15,5) dan judul grafik 'Audience score %'. Berikut tampilan dari perintah diatas.



Gambar 2.21 Grafik batang 'Audience score %'

2.2 NumPy

NumPy atau Numerical Python merupakan library pada Python yang digunakan untuk komputasi ilmiah. Pada sub bab ini, akan dibahas visualisasi data menggunakan NumPy, Matplotlib dan Seaborn. Matplotlib berupa grafik dasar yang meliputi grafik garis, batang, pie, dan scatter. Seaborn berupa grafik yang lebih kompleks seperti distributional plot. Pertama yang harus dilakukan untuk memulai suatu project menggunakan library NumPy dan Pandas yaitu sebagai berikut.

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

Jika akan melakukan visualisasi data menggunakan matplotlib dan seaborn perlu import library matplotlib dan seaborn itu sendiri, seperti berikut.

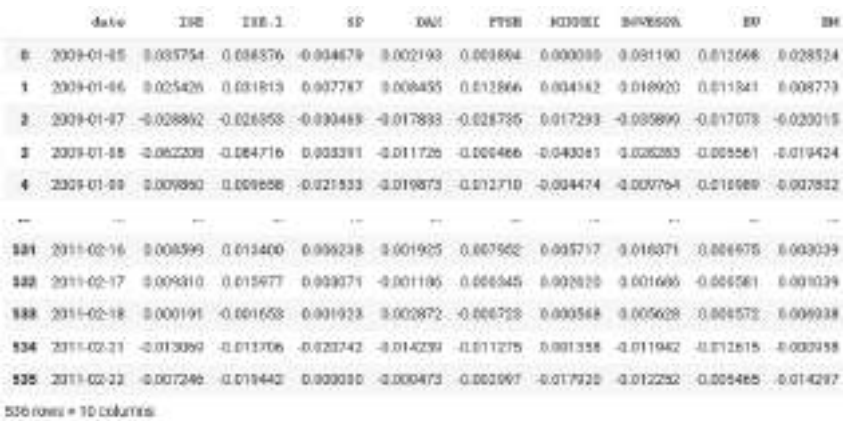
```
import matplotlib.pyplot as plt
import seaborn as sns
```

2.2.1 Import Berkas Excel

Import berkas excel dapat dilakukan menggunakan fungsi pd.read_excel seperti berikut.

```
stok_data = pd.read_excel('https://github.com/ammishra08/MachineLearning/blob/master/Datasets/data_akbilgic.xlsx?raw=true', header = 1)
```

Berikut isi dari data excel yang telah diimport.



	date	ISE	ISE.1	SP	DAX	FTSE	NIKKEI	BOVESPA	EU	EM
0	2009-01-05	0.035754	0.036576	-0.004679	0.002193	0.003894	0.000000	0.091190	0.012668	0.028524
1	2009-01-06	0.025426	0.031815	0.007787	0.008455	0.012866	0.004162	0.018920	0.011841	0.008773
2	2009-01-07	-0.028862	-0.026958	-0.000469	-0.017833	-0.028735	0.017293	-0.035899	-0.017078	-0.020015
3	2009-01-08	-0.062208	-0.064716	0.003391	-0.011726	-0.000466	-0.040361	0.026283	0.005561	-0.019424
4	2009-01-09	0.009860	0.009658	-0.021533	-0.019873	0.012710	-0.004474	-0.009764	0.010989	0.007832
...	...	...	...	...	...	...	...	...	...	...
531	2011-02-16	0.008599	0.013400	0.006238	0.001925	0.007662	0.005717	0.016071	0.006975	0.003039
532	2011-02-17	0.009310	0.015477	0.000071	-0.001186	0.006345	0.002820	0.001666	-0.000561	0.001039
533	2011-02-18	0.000191	-0.001658	0.001923	0.002872	-0.000723	0.000568	0.005628	0.000572	0.006938
534	2011-02-21	-0.013899	-0.013706	-0.020742	-0.014239	-0.011275	0.001358	-0.011942	-0.012615	-0.003958
535	2011-02-22	-0.007246	-0.010442	0.000000	-0.000473	0.002097	-0.017920	-0.012252	-0.005465	-0.014297

536 rows × 10 columns

Gambar 2.22 Data excel yang telah diimport

2.2.2 Matplotlib

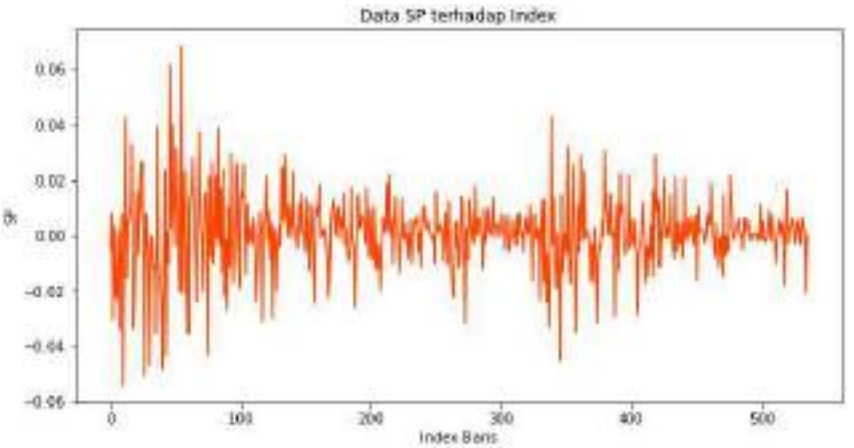
Pada subbab ini akan menjelaskan implementasi matplotlib untuk menampilkan grafik garis, batang, histogram, scatterplot.

2.2.2.1 Grafik Garis

Grafik garis menggunakan matplotlip dapat digunakan untuk visualisasi suatu data. Bertikut contoh penggunaan matplotlib.

```
plt.figure(figsize=(10,5))
plt.plot(stok_data['SP'], color = 'orangered')
plt.title("Data SP terhadap Index")
plt.xlabel("Index Baris")
plt.ylabel("SP")
plt.show()
```

SP merupakan data yang digunakan untuk nilai pada sumbu y dan x merupakan nilai dari index data. Berikut tampilan grafik garis menggunakan matplotlib.

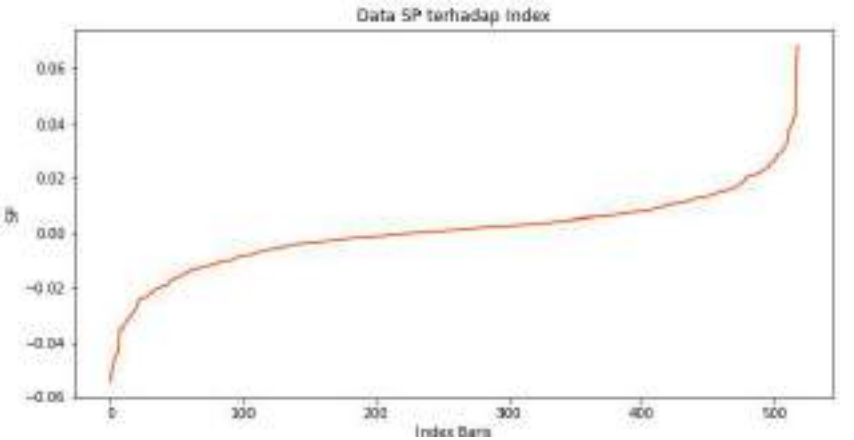


Gambar 2.23 Grafik garis menggunakan matplotlib

Data pada grafik dapat ditampilkan secara ascending menggunkan perintah berikut.

```
plt.figure(figsize=(10,5))
plt.plot(stok_data['SP'].sort_values(ascending =
True).unique(), color = 'orangered')
plt.title("Data SP terhadap Index")
plt.xlabel("Index Baris")
plt.ylabel("SP")
plt.show()
```

Berikut hasil grafik setelah diatur secara asceding.



Gambar 2.24 Grafik garis menggunakan matplotlib yang diatur ascending

2.2.2.2 Grafik Batang

Untuk menampilkan grafik batang juga dapat mengunakan matplotlib. Pertama yang dilakukan yaitu import berkas yang akan digunakan seperti berikut.

```
boston_df = pd.read_csv('https://raw.githubusercontent.com/ammishra08/MachineLearning/master/Datasets/boston_train.csv')
```

Berikut isi dari berkas csv diatas.

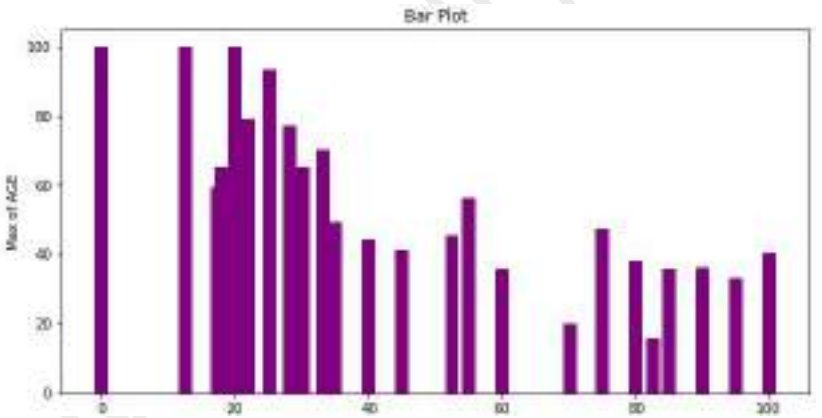
	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	TAX	PTRATIO	MEDV
0	2.30040	0.0	19.58	0.605	6.319	96.1	2.1000	403	14.7	23.8
1	13.35980	0.0	18.10	0.693	5.887	94.7	1.7821	666	20.2	12.7
2	0.12744	0.0	6.91	0.448	6.770	2.9	5.7209	233	17.9	26.6
3	0.15876	0.0	10.81	0.413	5.961	17.5	5.2873	305	19.2	21.7
4	0.03768	80.0	1.52	0.404	7.274	38.3	7.3090	329	12.6	34.6

Gambar 2.25 Data berkas boston_train.csv

Untuk menampilkan grafik batang dapat menggunakan fungsi plt.bar(). Berikut perintah selengkapnya.

```
plt.figure(figsize=(10,5))
plt.bar(x = boston_df['ZN'], height = boston_df['AGE'
], width=2, color = 'Purple')
plt.title("Bar Plot")
plt.xlabel("ZN")
plt.ylabel("Max of AGE")
plt.show()
```

x merupakan nilai pada sumbu x dimana data yang digunakan yaitu ZN dan heigh merupakan nilai pada sumbu y dimana data yang digunakan yaitu AGE. Berikut hasil grafiknya.



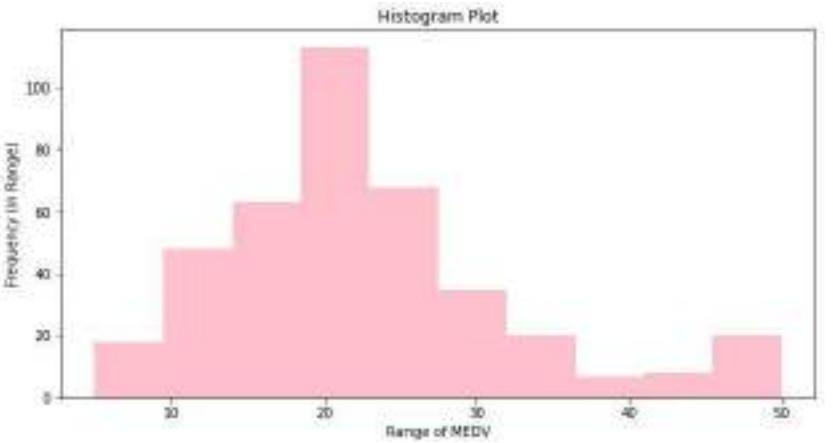
Gambar 2.26 Grafik batang menggunakan matplotlib

2.2.2.3 Grafik Histogram

Untuk menampilkan grafik histogram dapat menggunakan plt.hist(). Berikut perintah selengkapnya.

```
plt.figure(figsize = (10,5))
plt.hist(boston_df['MEDV'] ,color = 'Pink')
plt.title("Histogram Plot")
plt.xlabel("Range of MEDV")
plt.ylabel("Frequency (in Range)")
plt.show()
```

Berikut hasil grafik histogram.



Gambar 2.27 Grafik histogram menggunakan matplotlib

2.2.2.4 Grafik Scatterplot

Berkas yang digunakan untuk menampilkan grafik scatterplot menggunakan berkas csv iris.csv. Berikut berkas iris.csv yang akan digunakan.

```
iris_df = pd.read_csv('https://raw.githubusercontent.com/ammishra08/MachineLearning/master/Datasets/iris.csv', sep = ',')
```

Data dari berkas iris.csv yaitu sebagai berikut.

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

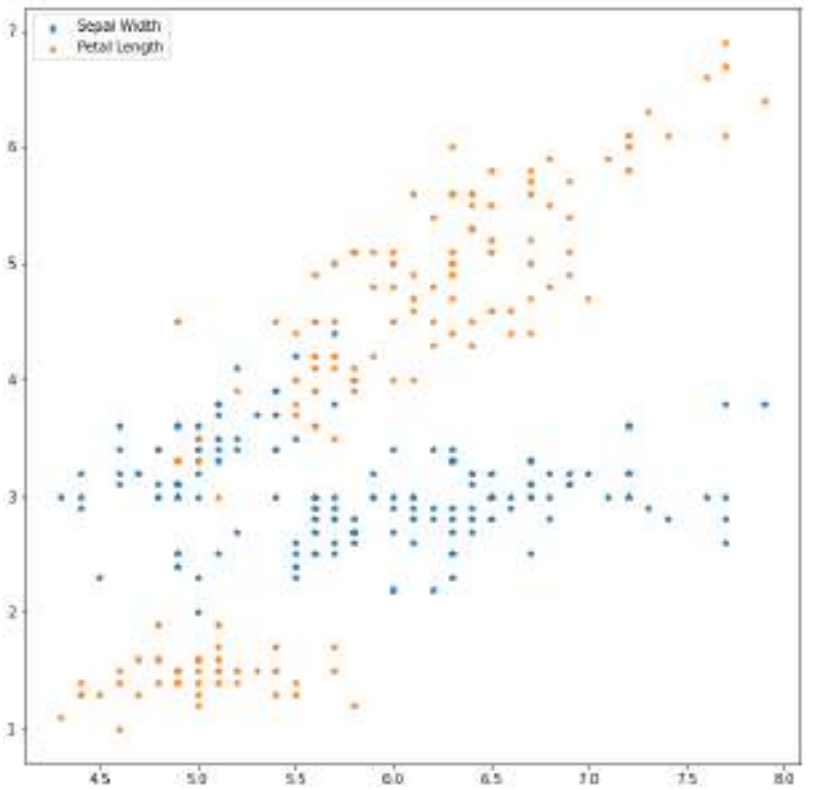
150 rows x 5 columns

Gambar 2.28 Data berkas iris.csv

Menampilkan grafik scatterplot dapat menggunakan fungsi plt.scatter. Berikut perintah selengkapnya.

```
plt.figure(figsize=(10,10))
plt.scatter(x = iris_df['sepal_length'], y = iris_d
f['sepal_width'], marker = '*', s = 25, label = 'Se
pal Width')
plt.scatter(x = iris_df['sepal_length'], y = iris_d
f['petal_length'], marker = '*', s = 25, label = 'P
etal Length')
plt.legend()
plt.show()
```

Berikut tampilan grafik scatterplot berdasarkan perintah diatas.



Gambar 2.29 Grafik scatterplot menggunakan matplotlib

'sepal_length' merupakan data yang digunakan untuk sumbu x. 'sepal_width' merupakan data yang digunakan untuk sumbu y pada scatter pertama. 'petal_length' merupakan data yang digunakan untuk sumbu y pada scatter kedua. Marker merupakan simbol yang digunakan untuk penanda data dan s merupakan ukuran dari marker yang digunakan. Label digunakan untuk memberikan keterangan maker.

2.2.3 Seaborn

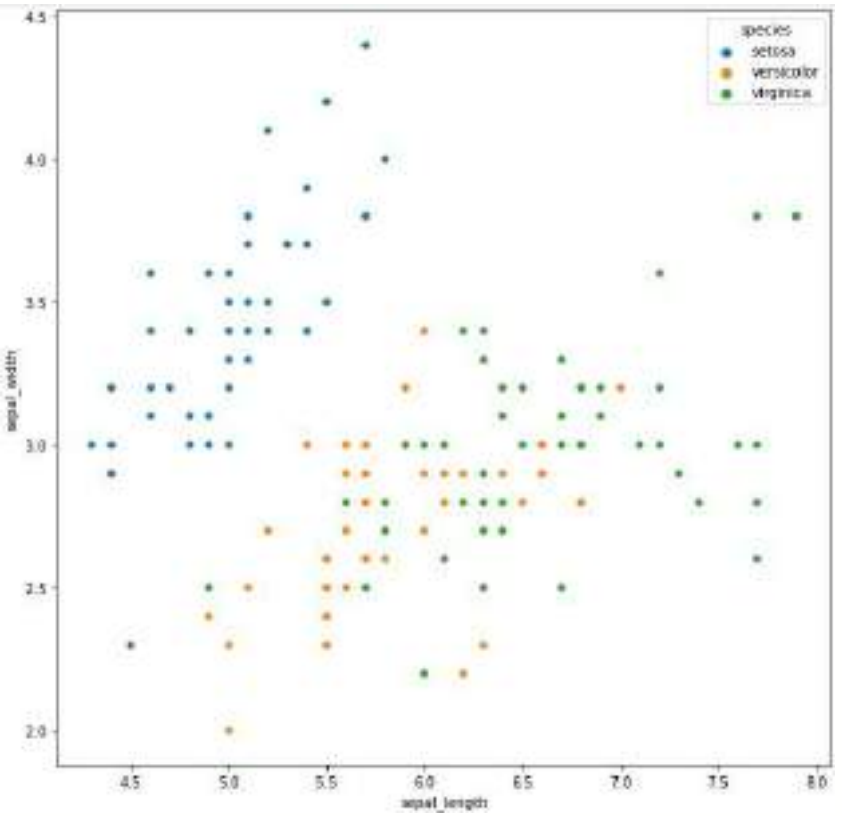
Pada seaborn akan dibahas visualisasi data menggunakan scatterplot, Pie, dan Distributional Plot.

2.2.3.1 Scatterplot

Untuk menampilkan scatterplot menggunakan seaborn, dapat menggunakan fungsi sns.scatterplot(). Berikut perintah selengkapnya.

```
plt.figure(figsize=(13,10))

sns.scatterplot(x = 'sepal_length', y = 'sepal_width
', data = iris_df, hue = 'species')
```

Berikut tampilan grafik scatterplot menggunakan seaborn.



Gambar 2.30 Grafik scatterplot menggunakan seaborn

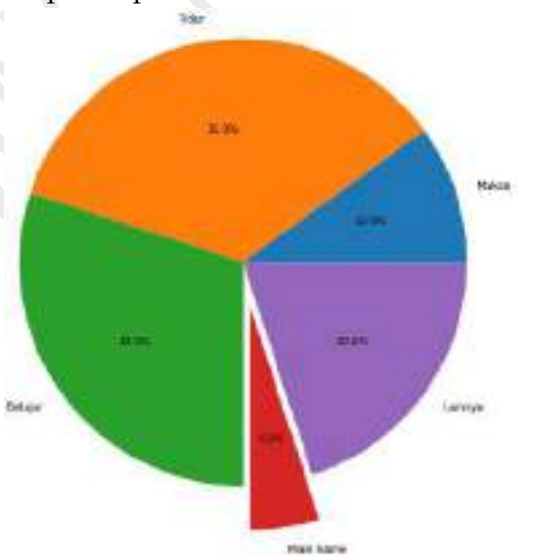
Hue merupakan data yang akan ditampilkan pada grafik. Data yang digunakan yaitu data species berdasarkan 'sepal_length' dan 'sepal_width'.

2.2.3.2 Pie

Grafik pie merupakan visualisasi data yang berbentuk lingkaran yang tediri dari irisan-irisan untuk mewakili suatu data. Untuk membuat grafik pie, fungsi yang digunakan yaitu plt.pie(). Berikut perintah selengkapnya.

```
slices = [2, 7, 6, 1, 4]
Aktifitas = ['Makan', 'Tidur', 'Belajar', 'Main Gam
e', 'Lainnya']

plt.figure(figsize=(10, 10))
plt.pie(slices, labels = Aktifitas, autopct='%1.1f%
 %', explode=(0,0,0,0.2,0))
plt.show()
```

Grafik pie ditampilkan pada Gambar 2.31.



Gambar 2.31 Grafik pie menggunakan seaborn

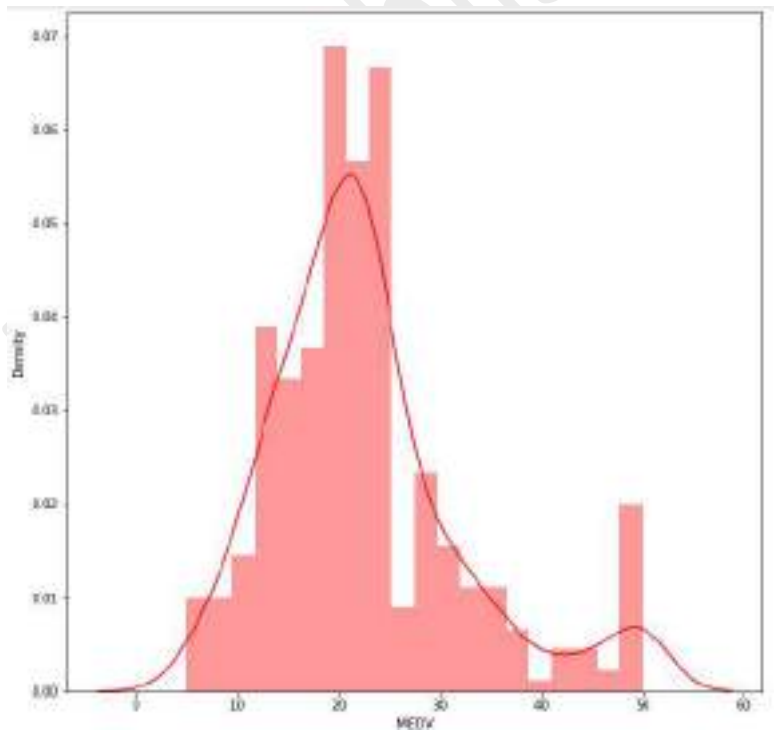
Slice merupakan ukuran dari masing-masing parameter aktifitas. Aktifitas merupakan label yang digunakan. Explode merupakan jarak antar bagian-bagian grafik.

2.2.3.3 Distributional Plot

Distributional plot digunakan untuk menampilkan grafik histogram dengan baris diatasnya. Untuk menampilkan grafiknya dapat menggunakan fungsi sns.distplot(). Berikut perintah lengkapnya.

```
plt.figure(figsize = (10,10))
sns.distplot(boston_df['MEDV'], color = 'red')
plt.show()
```

Grafik distributional plot ditampilkan pada Gambar 2.32.



Gambar 2.32 Grafik distributional plot

BAB III

PENERAPAN PYTHON UNTUK MEMBUAT CHATBOT

3.1 Definisi Chatbot

Chatbot atau chatting robot yang adalah program yang menyerupai percakapan dengan manusia menggunakan Artificial Intelligence (AI). Chatbot adalah agen percakapan otomatis yang berinteraksi dengan pengguna menggunakan bahasa alami manusia yang dapat membantu kapan saja dan dimana saja. Saat ini, dengan perkembangan kecerdasan buatan, chatbots digunakan di banyak bidang, seperti sistem penjawab telepon otomatis, dukungan pendidikan, bisnis, e-commerce, asisten virtual utama, tujuan hiburan, membantu seseorang menyelesaikan tugas mulai dari menjawab pertanyaan, mendapatkan petunjuk arah mengemudi, menyalakan termostat di rumah pintar, dan memainkan lagu favorit. Baru-baru ini, chatbot telah mendapat banyak perhatian dari para peneliti menunjukkan bahwa banyak penelitian telah dilakukan, seperti chatbot untuk menjawab Frequently Asked Questions (FAQ), aplikasi chatbot untuk pendidikan, atau evaluasi platform chatbot.

Chatbots merupakan aspek positif yang sudah lama digunakan sebagai agen pedagogis dalam pengaturan pendidikan. Sejak awal 1970-an, agen pedagogis dalam lingkungan pembelajaran digital yang dikenal sebagai Sistem Bimbingan Cerdas telah dikembangkan. Agen pedagogis percakapan menggunakan teknik kecerdasan buatan untuk meningkatkan otomatisasi dalam pengajaran. Pengetahuan desain dan penelitian penting dalam mengembangkan agen pedagogis yang menarik dan berguna yang tidak hanya memanfaatkan kemajuan teknologi, tetapi juga memahami masalah emosional, kognitif, dan pendidikan sosial.

Selain itu, agen percakapan telah dibangun ke dalam perangkat lunak dan perangkat keras.

Penggabungan chatbot ke dalam area pendidikan selama dekade terakhir menyiratkan peningkatan minat terhadap cara-cara penerapan chatbot untuk pengajaran dan pembelajaran. Sistem chatbot yang berguna dapat memberikan manfaat ketersediaan dengan instan dan kemampuan untuk merespons secara alami melalui antarmuka percakapan dengan keuntungan yang sama seperti wawancara. Selain itu, chatbot mendemonstrasikan kemampuan untuk menciptakan interaksi dengan pengguna sehingga dapat dimanfaatkan untuk mendukung keterlibatan, serta menetapkan tujuan, strategi, dan hasil pembelajaran dan pelatihan.

Perkembangan teknologi dan gaya hidup memungkinkan pembelajaran baru Bahasa Inggris yang dilakukan secara online (chat) melalui media sosial. Komunikasi bisa dalam bentuk teks (text chat) atau suara (voice chat) sehingga seolah-olah pengguna sedang berbicara dengan tutor. Keunggulan sistem ini adalah pengguna yang perlu berlatih bahasa Inggris dapat dilayani hampir 24 jam sehari sehingga tidak ada mendukung potensi interaksi siswa ke siswa dan siswa ke instruktur dengan menggunakan jejaring sosial.

3.2 Chatbot Sederhana Menggunakan Python

3.2.1 Install dan Import Library untuk ChatBot

Untuk membuat chatbot menggunakan Python, terlebih dahulu perlu install dan import library yang dibutuhkan, seperti chatterbot. Chatterbot merupakan library yang memudahkan menghasilkan tanggapan otomatis dari pertanyaan yang diajukan oleh pengguna. Pertama install chatterbot menggunakan command Python pip.

```
pip install chatterbot
pip install --upgrade chatterbot
```

Kemudian import ChatBot.

```
from chatterbot import ChatBot
```

3.2.2 Membuat Chatbot baru

Membuat chatbot baru dan pada contoh kali ini chatbot bernama "Shinta".

```
bot = ChatBot('Shinta')
```

3.2.3 Mengatur Adaptor Penyimpanan

ChatterBot memiliki kelas adaptor bawaan yang memungkinkan terhubung dengan berbagai jenis database. Dalam tutorial ini menggunakan SQLStorageAdapter yang memungkinkan chatbot terhubung dengan database SQL. Adaptor ini akan membuat database SQLite secara default. Parameter database digunakan untuk menentukan jalur ke database yang akan digunakan bot obrolan. Untuk contoh ini kita akan memanggil database 'sqlite:///database.sqlite3. file ini akan dibuat secara otomatis.

```
bot = ChatBot(
    'Shinta',
    storage_adapter='chatterbot.storage.SQLStorage
Adapter',
    database_uri='sqlite:///database.sqlite3'
)
```

3.2.4 Menentukan Adaptor Logika

Parameter logic_adapters adalah daftar adaptor logika. Adaptor logika pada chatterbot adalah kelas yang mengambil pernyataan input dan mengembalikan respons ke pernyataan tersebut. Adaptor logika yang digunakan bisa banyak sesuai yang dibutuhkan. Dalam contoh ini adapter logika yang digunakan yaitu TimeLogicAdapter dan MathematicalEvaluation.

TimeLogicAdapter mengembalikan waktu saat ini ketika pernyataan input memintanya. Adaptor MathematicalEvaluation memecahkan masalah matematika yang menggunakan operasi dasar.

```
bot = ChatBot(
    'Shinta',
    logic_adapters=[
        'chatterbot.logic.MathematicalEvaluation',
        'chatterbot.logic.TimeLogicAdapter'],
database_uri='sqlite:///database.sqlite3'
)
```

3.2.5 Training Chatbot

Pada proses training ini Shinta akan belajar berkomunikasi saat pengguna berbicara dengan bot menggunakan modul yang tersedia di chatterbot. Pertama, impor ListTrainer, buat objeknya dengan meneruskan objek Chatbot. ListTrainer digunakan untuk mengizinkan bot obrolan untuk dilatih menggunakan daftar string di mana daftar tersebut mewakili percakapan. Kemudian dilanjutkan dengan train() dengan meneruskan daftar kalimat.

```
from chatterbot.trainers import ListTrainer

trainer = ListTrainer(bot)
```

```
trainer.train([
'Hi',
'Hello',
'Apa Kabar?',
'Baik, Bagaimana kabarmu?',
'Baik Juga, Terima kasih',
'Oke, Sama-sama',
'Jawab penjumlahan berikut'
'Oke.'
'1+1=',
'2',
'1+2=',
'3',
'1+3=',
'4',
'2+3=',
'5',
'3+3=',
'4'
])
```

3.2.6 esting Chatbot

Langkah terakhir dari tutorial ini adalah menguji keterampilan percakapan chatterbot. Untuk menguji tanggapannya dapat dilakukan dengan memanggil metode get_response() dari instance Chatbot.

```
response = bot.get_response('I have a complaint.')

print("Bot Response:", response)
```

Kemudian membuat while loop untuk menjalankan chatbot. Ketika pernyataan dilewatkan dalam loop, kita akan mendapatkan respon yang sesuai, karena kita telah memasukkan data ke database kita. Jika mendapatkan pernyataan "Selamat Tinggal" dari pengguna, kita dapat mengakhiri loop dan menghentikan program.

```
name=input("Enter Your Name: ")
print("Selamat datang di layanan Chatbot!")
while True:
    request=input(name+':')
    if request=='Bye' or request =='Selamat Tinggal':
        print('Bot: Selamat Tinggal')
        break
    else:
        response=bot.get_response(request)
        print('Bot:',response)
```

dan berikut percakapannya.

```
Enter Your Name: Cici
Selamat datang di layanan Chatbot!
Cici:Hi
Bot: Hello
Cici:Apa Kabar?
Bot: Baik, Bagaimana kabarmu?
Cici:Baik Juga, Terima Kasih
Bot: Oke, Sama-sama
Cici:Jawab penjumlahan berikut
Bot: The current time is 03:44 PM
Cici:1+1=
Bot: 2
Cici:2+3=
Bot: 5
Cici:1+3=
Bot: 4
Cici:Selamat Tinggal
Bot: Selamat Tinggal
```

3.3 Dasar Membuat Chatbot Sederhana Menggunakan Media Facebook Messenger

Beberapa tahapan yang dilakukan untuk membuat Chatbot menggunakan Facebook Messenger yaitu sebagai berikut.

3.3.1. Membuat aplikasi pada Facebook Developer

Sebelum membangun Chatbot pada Python, perlu membuat aplikasi pada Facebook Developer terlebih dahulu dan melakukan pengaturan-pengaturan seperti yang telah dijelaskan pada halaman Facebook Developer.

3.3.2. Merancang dan membuat database

Setelah membuat aplikasi pada Facebook, tahapan selanjutnya yaitu merancang dan membuat database dengan table-tabel sesuai yang dibutuhkan. Database yang digunakan pada buku ini yaitu database MySQL.

3.3.3. Import library yang dibutuhkan

Tahapan pertama yang dilakukan pada Python yaitu import *library* seperti random, flask, json, request, dan islice. Import beberapa library tersebut dapat dilakukan dengan script berikut.

```
import random
from flask import Flask, request
import json
import requests
```

Beberapa library yang digunakan yaitu:

1. **Random** merupakan *library* untuk melakukan operasi-operasi yang berkaitan dengan bilangan acak untuk menentukan suatu pilihan.
2. **Flask** merupakan kerangka kerja yang menyediakan tools, library, dan teknologi untuk membangun halaman web dengan Bahasa pemrograman Python.
3. **Json** merupakan *syntax* yang digunakan untuk menyimpan dan bertukar data.
4. **Requests** merupakan *library* HTTP yang ditulis dengan Bahasa pemrograman Python. Sehingga tidak perlu menambahkan string query secara manual ke URL.

3.3.4. Membuat instance Flask

Membuat instance Flask dengan script berikut.

```
app = Flask(__name__)
```

3.3.5. Menghubungkan Python dengan Facebook Messenger

Python dan Facebook Messenger dapat dihubungkan dengan menggunakan access token dan verify token. ACCESS_TOKEN diperoleh dari pengaturan pada Facebook developer. Sedangkan VERIFY_TOKEN dibuat sendiri oleh developer aplikasi yang nantinya dimasukkan pada pengaturan Facebook developer.

```
ACCESS_TOKEN =
'EAAHdVUDEWtIBAEICjZCceKI6L2OQKJBRe2XMbMTP8gSpFmJN
1nKl4FCH5hQZAxQPxOmOYxVzK7BLPN8JLgiKDQBM7XicWEu294
G9pZAIEptY2MQfsvP6o8DZBoyGSq2RA9ktrpu1DTofUsEwZAVL
9ZB8lUx8U0zauS'
VERIFY_TOKEN = 'token'
```

3.3.6. Menerima pesan dari Facebook

Menerima pesan dari Facebook yang dikirim oleh pengguna melalui Facebook Messenger dapat dilakukan menggunakan program berikut

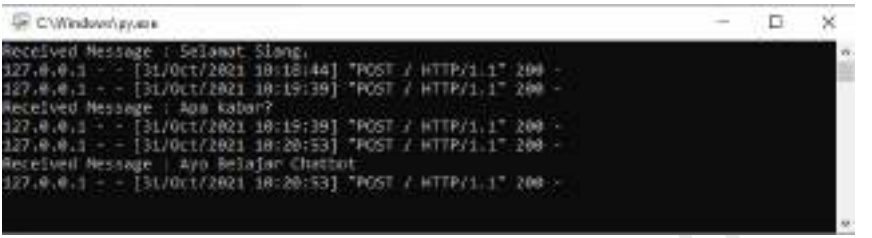
```
def handleMessage(senderPsid, receivedMessage):
    print("Received Message : " + str(receivedMessage))
    if (receivedMessage == 'start' or receivedMessage
== 'Start'):
         print("==============START
CHATBOT==============")

@app.route("/", methods=['GET', 'POST'])
def webhook():
    if request.method == 'GET':
        if request.args.get("hub.verify_token") ==
VERIFY_TOKEN:
            return request.args.get("hub.challenge")
        else:
            return "Not Connected"
```

```
    elif request.method == 'POST':
        data = request.data
        body = json.loads(data.decode('utf-8'))
        if 'object' in body and body['object'] ==
'page':
            entries = body['entry']
            for entry in entries:
                webhookEvent = entry['messaging'][0]
                senderPsid =
webhookEvent['sender']['id']
                if 'message' in webhookEvent:
                    webhookEventMessage =
webhookEvent['message']
                    if 'is_echo' in
webhookEventMessage:
                        return '1'
                    else:
                        handleMessage(senderPsid,
webhookEventMessage['text'])
                return 'EVENT_RECEIVED', 200
        return '1'
    else:
        return "200"

if __name__ == "__main__":
    app.run()
```

Berikut tampilan pada Facebook Messenger ketika mengirim pesan.

Gambar 3.1 Tampilan mengirim pesan pada Facebook Messenger

Berikut tampilan pada Python yang telah menerima pesan dari Facebook Messenger. Pesan yang diterima oleh Python dapat dilihat pada command yang tampil ketika program Python dijalankan.



Gambar 3.2 Tampilan Python menerima pesan dari Facebook Messenger

3.3.7. Mengirim pesan ke Facebook

Python mengirim pesan ke Facebook Messenger dapat berupa pesan teks, gambar, dan audio. Berikut program yang digunakan untuk mengirim pesan teks, gambar, audio, dan membuat pilihan.

```
from flask import Flask, request
import json
import requests

app = Flask(__name__)

ACCESS_TOKEN =
'EAAEJc2YfZBK8BAOLvQhOI0bbOjbJYHz5ZBUYEOSN7ZBqt5xlS
WmVEgIrMDBBlkw5ufemiwbC05L5tZBdm0QTUJxjdFZBNb4AKlb9
j6DZByALYajJroZCq4pRpZA2qjRMybLqZA82syecBLCpypJNyy4
pfk7trQUQI8uhZAJ7thyO1EhhEFwfqueEJZB'
VERIFY_TOKEN = 'tokenbot'

def sendAudio(senderPsid):
    payload = {
        "messaging_type": "RESPONSE",
        "recipient":{
            "id": senderPsid
        },
        "message":{
            "attachment":{
                "type":"audio",
                "payload":{
                    "url":"https://file-examples-
com.github.io/uploads/2017/11/file_example_MP3_700K
B.mp3"
```

```
                }
            }
        }
    }
    headers = {'content-type' : 'application/json'}
    url =
'https://graph.facebook.com/v11.0/me/messages?access_
token='+ACCESS_TOKEN
    r = requests.post(url, json=payload,
headers=headers)

def sendOption(senderPsid):
    payload = {
        "messaging_type": "RESPONSE",
        "recipient": {
            "id": senderPsid
        },
        "message": {
            "text": "Ingin Melakukan Latihan Apa Hari
Ini ?",
            "quick_replies": [
                {
                    "content_type": "text",
                    "title": "Opsi 1",
                    "payload": "value_1",

                },
                {
                    "content_type": "text",
                    "title": "Opsi 2",
                    "payload": "value_2",
                },
                {
                    "content_type": "text",
                    "title": "Opsi 3",
                    "payload": "value_3",
                }
            ]
        }
    }
```

```
    headers = {'content-type': 'application/json'}
    url =
'https://graph.facebook.com/v11.0/me/messages?access_t
oken='+ACCESS_TOKEN
    r = requests.post(url, json=payload,
headers=headers)

def sendMessage(senderPsid, response):
    payload = {
        "messaging_type": "RESPONSE",
        "recipient": {
            "id": senderPsid
        },
        "message": {
            "text": response,
        }
    }

    headers = {'content-type': 'application/json'}
    url =
'https://graph.facebook.com/v11.0/me/messages?access_t
oken='+ACCESS_TOKEN
    r = requests.post(url, json=payload,
headers=headers)

def sendImage(senderPsid):
    payload = {
        "messaging_type": "RESPONSE",
        "recipient":{
            "id": senderPsid
        },
        "message":{
            "attachment":{
                "type":"image",
                "payload":{
                    "url":"https://download-free-
images.com/00002/cute-gif-for-you-white-little-bird-
176256.gif"
                }
            }
        }
    }
```

```
    headers = {'content-type' : 'application/json'}
    url =
'https://graph.facebook.com/v11.0/me/messages?access_to
ken='+ACCESS_TOKEN
    r = requests.post(url, json=payload,
headers=headers)

def handleMessage(senderPsid, receivedMessage):
    print("Received Message : " + str(receivedMessage))
    if (str(receivedMessage) == 'Audio'):
        sendAudio(senderPsid)
    elif (str(receivedMessage) == 'Text'):
        sendMessage(senderPsid, "Ini Adalah Balasan
dari Sistem Chatbot")
    elif (str(receivedMessage) == 'Image'):
        sendImage(senderPsid)
    elif (receivedMessage == 'Option'):
        sendOption(senderPsid)

@app.route("/", methods=['GET', 'POST'])
def webhook():
    if request.method == 'GET':
        if request.args.get("hub.verify_token") ==
VERIFY_TOKEN:
            return request.args.get("hub.challenge")
        else:
            return "Not Connected"
    elif request.method == 'POST':
        data = request.data
        body = json.loads(data.decode('utf-8'))
        if 'object' in body and body['object'] ==
'page':
            entries = body['entry']
            for entry in entries:
                webhookEvent = entry['messaging'][0]
                senderPsid =
webhookEvent['sender']['id']
                if 'message' in webhookEvent:
                    webhookEventMessage =
webhookEvent['message']
                    if 'is_echo' in
webhookEventMessage:
                        return '1'
                    else:
                        handleMessage(senderPsid,
webhookEventMessage['text'])
```

```
                return 'EVENT_RECEIVED', 200
        return '1'
    else:
        return "200"

if __name__ == "__main__":
    app.run()
```

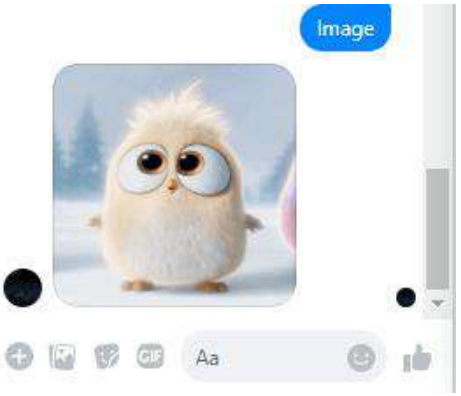
Berikut tampilan pada Facebook Messenger ketika menerima pesan teks dari Python. Ketika pengguna mengirim pesan "Text", Python dapat memberikan respon kepada pengguna berupa pesan teks.

Gambar 3.3 Tampilan Facebook Messenger ketika menerima pesan teks dari Python

Kemudian pada command Python tampil seperti pada gambar dibawah ketika menerima pesan dari pengguna.

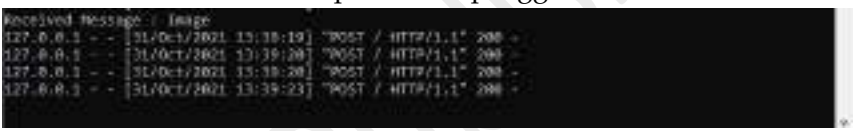
Gambar 3.4 Tampilan Python menerima pesan untuk menampilkan pesan teks

Berikut tampilan pada Facebook Messenger ketika menerima pesan gambar dari Python. Python akan mengirimkan pesan gambar ketika pengguna mengirimkan pesan "Image".



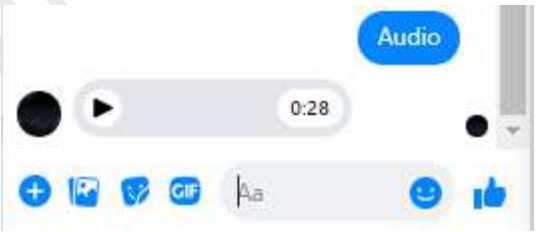
Gambar 3.5 Tampilan Facebook Messenger ketika menerima pesan gambar dari Python

Kemudian pada command Python tampil seperti pada gambar dibawah ketika menerima pesan dari pengguna.



Gambar 3.6 Tampilan Python menerima pesan untuk menampilkan pesan gambar

Berikut tampilan pada Facebook Messenger ketika menerima pesan audio dari Python. Python akan mengirimkan pesan audio ketika pengguna mengirimkan pesan "Audio".



Gambar 3.7 Tampilan Facebook Messenger ketika menerima pesan audio dari Python

Berikut tampilan pada Python ketika mengirim pesan audio.

Gambar 3.8 Tampilan Python menerima pesan untuk menampilkan pesan audio

Ketika pengguna mengirimkan pesan "Option". Maka Python akan mengirimkan pesan berupa pilihan. Pada contoh berikut terdapat tiga pilihan yang di berikan oleh Python yaitu "Opsi 1", "Opsi 2", dan "Opsi 3". Berikut tampilan pada Facebook Messenger.

Gambar 3.9 Tampilan Facebook Messenger ketika menerima pesan untuk menampilkan pilihan

Berikut tampilan pada Python ketika mengirim pesan yang berupa pilihan.

Gambar 3.10 Tampilan Python menerima pesan untuk menampilkan pesan audio

3.4 Chatbot untuk Pembelajaran Bahasa Inggris Menggunakan Media Facebook Messenger

Pada subbab ini akan dijelaskan cara membuat suatu sistem Chatbot untuk pembelajaran Bahasa inggris menggunakan media Facebook Messenger. Pada sistem ini berisi latihan soal dengan jawaban pilihan ganda. Latihan soal terdiri dari dari latihan

structure, reading, dan listening. Sehingga pengguna dapat memilih jenis soal latihan yang diinginkan. Masing-masing jenis soal memiliki tingkat kesulitan yang berbeda yang terdiri dari tiga tingkatan yaitu Level 1, Level 2, dan Level 3. Level 1 memiliki tingkat kerumitan soal yang paling mudah dan Level 3 merupakan tingkatan yang paling sulit. Pengguna baru akan memulai latihan dari pretest untuk menentukan Level latihan selanjutnya. Ketika pengguna mendapatkan nilai 1 sampai 2, maka pengguna akan mendapatkan latihan Level 1 pada latihan berikutnya. Jika mendapatkan nilai 3 dan 4, pengguna akan mendapatkan latihan Level 2, dan jika mendapatkan nilai 5, pengguna akan mendapatkan latihan Level 3 pada latihan berikutnya. Setelah melakukan latihan sesuai Level kemampuan pengguna, pengguna bisa melanjutkan ke posttest untuk menentukan level latihan berikutnya dengan ketentuan penilaian sama dengan pretest. Jadi, masing-masing pengguna bisa naik dan turun level. Data-data terkait soal-soal latihan, jawaban, dan data-data lain yang berkaitan dengan Chatbot disimpan di database MySQL yang telah dibuat sebelumnya. Berikut program yang digunakan untuk membuat sistem Chatbot untuk pembelajaran Bahasa inggris.

```
import random
from flask import Flask, request
import json
import requests
import mysql.connector
from itertools import islice

app = Flask(__name__)
```

```
ACCESS_TOKEN =
'EAAHdVUDEWtIBAEICjZCceKI6L2OQKJBRe2XMbMTP8gSpFmJN1nKl4
FCH5hQZAxQPxOmOYxVzK7BLPN85r5adjvZAa4nQDI2gWzrwJLgiKDQB
M7XicWEu294G9pZAIEptY2MQfsvP6o8DZB
VERIFY_TOKEN = 'tokenbot'

def connect():
    conn = mysql.connector.connect(
        host="localhost",
        user="admin",
        password="pass_admin",
        db="learn_english"
    )
    cur = conn.cursor()
    return cur, conn

def sendCategory(senderPsid):
    payload = {
        "messaging_type": "RESPONSE",
        "recipient": {
            "id": senderPsid
        },
        "message": {
            "text": "Ingin Melakukan Latihan Apa Hari
Ini?",
            "quick_replies": [
                {
                    "content_type": "text",
                    "title": "Structure",
                    "payload": "structure",
                },
                {
                    "content_type": "text",
                    "title": "Reading",
                    "payload": "reading",
                },
                {
                    "content_type": "text",
                    "title": "Listening",
                    "payload": "listening",
                }
```

```
            ]
        }
  }
    headers = {'content-type': 'application/json'}
    url =
'https://graph.facebook.com/v11.0/me/messages?access
_token=EAAHdVUDEWtIBAEICjZCceKI6L2OQKJBRe2XMbMTP8gSp
FmJN1nKl4FCH5hQZAxQPxOmOYxVzK7BLPN85r5adjvZAa4nQDI2g
WzrwJLgiKDQBM7XicWEu294G9pZAIEptY2MQfsvP6o8DZBoyGSq2
RA9ktrpu1DTofUsEwZAVL9ZB8lUx8U0zauS'
    r = requests.post(url, json=payload,
headers=headers)

def sendConfirmationPostTest(senderPsid):
    payload = {
        "messaging_type": "RESPONSE",
        "recipient": {
            "id": senderPsid
        },
        "message": {
            "text": "Ingin Melanjutkan ke Post Test
?",
            "quick_replies": [
                {
                    "content_type": "text",
                    "title": "Iya",
                    "payload": "post_test_approved",
                },
                {
                    "content_type": "text",
                    "title": "Tidak",
                    "payload": "post_test_cancel",
                }
            ]
        }
    }
    headers = {'content-type': 'application/json'}
    url =
'https://graph.facebook.com/v11.0/me/messages?access
_token=EAAHdVUDEWtIBAEICjZCceKI6L2OQKJBRe2XMbMTP8gSp
FmJN1nKl4FCH5hQZAxQPxOmOYxVzK7BLPN85r5adjvZAa4nQDI2g
WzrwJLgiKDQBM7XicWEu294G9pZAIEptY2MQfsvP6o8DZBoyGSq2
RA9ktrpu1DTofUsEwZAVL9ZB8lUx8U0zauS'
    r = requests.post(url, json=payload,
headers=headers)
```

```
def stopConfirmation(senderPsid):
    payload = {
        "messaging_type": "RESPONSE",
        "recipient": {
            "id": senderPsid
        },
        "message": {
            "text": "Terdapat sesi yang belum selesai,
Anda ingin hentikan sesi sebelumnya ?",
            "quick_replies": [
                {
                    "content_type": "text",
                    "title": "Iya",
                    "payload": "stop_session",

                },
                {
                    "content_type": "text",
                    "title": "Tidak",
                    "payload": "continue_session",
                }
            ]
        }
    }

    headers = {'content-type': 'application/json'}
    url =
'https://graph.facebook.com/v11.0/me/messages?access_t
oken=EAAHdVUDEWtIBAEICjZCceKI6L2OQKJBRe2XMbMTP8gSpFmJN
1nKl4FCH5hQZAxQPxOmOYxVzK7BLPN85r5adjvZAa4nQDI2gWzrwJL
giKDQBM7XicWEu294G9pZAIEptY2MQfsvP6o8DZBoyGSq2RA9ktrpu
1DTofUsEwZAVL9ZB8lUx8U0zauS'
    r = requests.post(url, json=payload,
headers=headers)

def sendQuestionToUser(senderPsid, session_id,
question):
    cur, conn    = connect()
    query_delete = "DELETE FROM `questions` WHERE
`questions`.`question_id` = %s AND
`questions`.`session_id` = %s"
    cur.execute(query_delete,(question[0],
session_id,))
    conn.commit()
```

```
    payload = {
        "messaging_type": "RESPONSE",
        "recipient": {
            "id": senderPsid
        },
        "message": {
            "text": str(question[2]) + '\n' + "-----
----------" + "\n A. " + str(question[3]) + "\n B. "
+ str(question[4]) + "\n C. " + str(question[5]) +
"\n D. " + str(question[6]),

            "quick_replies": [
            {
                "content_type": "text",
                "title": "#A",
                "payload":session_id+"_"+str(questio
n[0])+"_"+str(question[3]),
            },

            {
                "content_type": "text",
                "title": "#B",
                "payload":session_id+"_"+str(questio
n[0])+"_"+str(question[4]),
            },

            {
                "content_type": "text",
                "title": "#C",
                "payload":session_id+"_"+str(questio
n[0])+"_"+str(question[5]),
            },

            {
                "content_type": "text",
                "title": "#D",
                "payload":session_id+"_"+str(questio
n[0])+"_"+str(question[6]),
            }]
        }
    }
```

```
    headers = {'content-type': 'application/json'}
    url =
'https://graph.facebook.com/v11.0/me/messages?access_toke
n=EAAHdVUDEWtIBAEICjZCceKI6L2OQKJBRe2XMbMTP8gSpFmJN1nKl4F
CH5hQZAxQPxOmOYxVzK7BLPN85r5adjvZAa4nQDI2gWzrwJLgiKDQBM7X
icWEu294G9pZAIEptY2MQfsvP6o8DZBoyGSq2RA9ktrpu1DTofUsEwZAV
L9ZB8lUx8U0zauS'
    r = requests.post(url, json=payload, headers=headers)

def sendQuestion(senderPsid, session_id, question_type):
    cur, conn = connect()
    query_question = "SELECT * FROM questions WHERE
session_id = %s"
    cur.execute(query_question,(session_id,))
    question_id = cur.fetchone()

    if (question_id == None):
        print("SELESAI")
        generateScore(senderPsid, session_id,
question_type)
    else:
        last_material_id = 0
        if (question_type == "structure"):
            cur, conn = connect()
            query_structure = "SELECT * FROM
`structure_question` WHERE id = %s LIMIT 1"
            cur.execute(query_structure,(str(question_id[
2]),))
            question = cur.fetchone()
            sendQuestionToUser(senderPsid, session_id,
question)
        elif (question_type == "reading"):
            cur, conn = connect()
            query_reading = "SELECT * FROM
`reading_question` WHERE id = %s LIMIT 1"
            cur.execute(query_reading,(str(question_id[2]
),))
            question = cur.fetchone()
            if (last_material_id != question[1]):
                last_material_id = question[1]
                query_material = "SELECT * FROM
`reading_material` WHERE id = %s LIMIT 1"
                cur.execute(query_material,(str(question[
1]),))
                material = cur.fetchone()
                sendMessage(senderPsid, str(material[2]))
            sendQuestionToUser(senderPsid, session_id,
question)
```

```
def sendMessage(senderPsid, response):
    payload = {
        "messaging_type": "RESPONSE",
        "recipient": {
            "id": senderPsid
        },
        "message": {
            "text": response,
        }
    }
    headers = {'content-type': 'application/json'}
    url =
'https://graph.facebook.com/v11.0/me/messages?acces
s_token=EAAHdVUDEWtIBAEICjZCceKI6L2OQKJBRe2XMbMTP8g
SpFmJN1nKl4FCH5hQZAxQPxOmOYxVzK7BLPN85r5adjvZAa4nQD
I2gWzrwJLgiKDQBM7XicWEu294G9pZAIEptY2MQfsvP6o8DZBoy
GSq2RA9ktrpu1DTofUsEwZAVL9ZB8lUx8U0zauS'
    r = requests.post(url, json=payload,
headers=headers)

def sendAudio(senderPsid, response):
    payload = {
        "messaging_type": "RESPONSE",
        "recipient":{
            "id": senderPsid
        },
        "message":{
            "attachment":{
                "type":"audio",
                "payload":{
                    "url":response
                }
            }
        }
    }
    headers = {'content-type' : 'application/json'}
    url =
'https://graph.facebook.com/v11.0/me/messages?acces
s_token=EAAHdVUDEWtIBAEICjZCceKI6L2OQKJBRe2XMbMTP8g
SpFmJN1nKl4FCH5hQZAxQPxOmOYxVzK7BLPN85r5adjvZAa4nQD
I2gWzrwJLgiKDQBM7XicWEu294G9pZAIEptY2MQfsvP6o8DZBoy
GSq2RA9ktrpu1DTofUsEwZAVL9ZB8lUx8U0zauS'
    r = requests.post(url, json=payload,
headers=headers)
```

```
def saveAnswer(senderPsid, answer):
    answer_val = answer.split("_")

    cur, conn = connect()
    query = "INSERT INTO `answers` (`id`,
`session_id`, `user_id`, `question_id`, `answer`)
VALUES (NULL, %s, %s, %s, %s)"
    cur.execute(query,(answer_val[0], senderPsid,
answer_val[1], answer_val[2],))
    conn.commit()

    session_id = checkSession(senderPsid)
    question_type = getQuestionType(session_id)
    sendQuestion(senderPsid, str(session_id),
str(question_type))


def generateScore(senderPsid, session_id,
question_type):
    score = 0

    cur, conn = connect()
    query_answer = "SELECT * FROM answers WHERE
session_id = %s AND user_id = %s"
    cur.execute(query_answer,(session_id,
senderPsid,))
    data_answer = cur.fetchall()
    for i in data_answer:
        if (question_type == "listening"):
            query_check = "SELECT answer FROM
listening_question WHERE id = %s"
            cur.execute(query_check,(i[3],))
            data_question = cur.fetchone()
            if (str(data_question[0]) == str(i[4])):

                score = score + 1
        elif (question_type == "reading"):
            query_check = "SELECT answer FROM
reading_question WHERE id = %s"
            cur.execute(query_check,(i[3],))
            data_question = cur.fetchone()

            if (str(data_question[0]) == str(i[4])):
                score = score + 1
```

```
        elif (question_type == "structure"):
            query_check = "SELECT answer FROM
structure_question WHERE id = %s"
            cur.execute(query_check,(i[3],))
            data_question = cur.fetchone()
            if (str(data_question[0]) == str(i[4])):
                score = score + 1

    query_update = "UPDATE `sessions` SET `score` =
%s, `status` = %s WHERE `sessions`.`id` = %s"
    cur.execute(query_update,(str(score),
"completed", session_id,))
    conn.commit()

    assesment_type = getAssesmentType(session_id)
    print("ASSESMENT : " + str(assesment_type))

    if (str(assesment_type) == "pre_test"):
        if (score <3):
            update_level = 1
        elif (score >= 3 and score < 5):
            update_level = 2
        elif (score >= 5):
            update_level = 3

        if (question_type == "listening"):
            query_update_level = "UPDATE `users` SET
`level_listening` = %s WHERE `users`.`id` = %s"
            cur.execute(query_update_level,(str(upda
te_level), senderPsid,))
        elif (question_type == "structure"):
            query_update_level = "UPDATE `users` SET
`level_structure` = %s WHERE `users`.`id` = %s"
            cur.execute(query_update_level,(str(upda
te_level), senderPsid,))
        elif (question_type == "reading"):
            query_update_level = "UPDATE `users` SET
`level_reading` = %s WHERE `users`.`id` = %s"
            cur.execute(query_update_level,(str(upda
te_level), senderPsid,))
        conn.commit()
```

```
    if (str(assesment_type) == "post_test"):
        current_level = checkLevel(senderPsid,
question_type)
        if (score <3):
            if (int(str(current_level)) == 1):
                update_level = int(str(current_level))
            else:
                update_level = int(str(current_level))
- 1
        elif (score >= 3 and score < 5):
            update_level = int(str(current_level))
        elif (score >= 5):
            if (int(str(current_level)) == 3):
                update_level = int(str(current_level))
            else:
                update_level = int(str(current_level))
+ 1

        if (question_type == "listening"):
            query_update_level = "UPDATE `users` SET
`level_listening` = %s WHERE `users`.`id` = %s"
            cur.execute(query_update_level,(str(update_
level), senderPsid,))
        elif (question_type == "structure"):
            query_update_level = "UPDATE `users` SET
`level_structure` = %s WHERE `users`.`id` = %s"
            cur.execute(query_update_level,(str(update_
level), senderPsid,))
        elif (question_type == "reading"):
            query_update_level = "UPDATE `users` SET
`level_reading` = %s WHERE `users`.`id` = %s"
            cur.execute(query_update_level,(str(update_
level), senderPsid,))
        conn.commit()

    sendMessage(senderPsid, "Score Anda : " +
str(score))
    if (str(assesment_type) == "pre_test"):
        sendMessage(senderPsid, "Terima Kasih Telah
Mengerjakan Soal Pre Test\nLevel Anda Saat Ini : " +
str(update_level))
    elif (str(assesment_type) == "post_test"):
        sendMessage(senderPsid, "Terima Kasih Telah
Mengerjakan Soal Post Test\nLevel Anda Saat Ini : " +
str(update_level))
    elif (str(assesment_type) == "exercise"):
        sendConfirmationPostTest(senderPsid)
```

```
def checkSession(senderPsid):
    cur, conn = connect()
    query = "SELECT * FROM sessions WHERE
`user_id` = %s AND `status` = 'processing' ORDER
by id DESC LIMIT 1"
    cur.execute(query, (senderPsid,))
    data = cur.fetchone()
    if (data == None):
        return 0
    else:
        return data[0]

def getQuestionType(session_id):
    cur, conn = connect()
    query = "SELECT * FROM sessions WHERE `id` =
%s"
    cur.execute(query, (session_id,))
    data = cur.fetchone()
    if (data == None):
        return 0
    else:
        return data[4]

def getAssesmentType(session_id):
    cur, conn = connect()
    query = "SELECT * FROM sessions WHERE `id` =
%s"
    cur.execute(query, (session_id,))
    data = cur.fetchone()
    if (data == None):
        return 0
    else:
        return data[5]

def checkLevel(senderPsid, question_type):
    cur, conn = connect()
    query = "SELECT * FROM users WHERE id = %s"
    cur.execute(query, (senderPsid,))
    data = cur.fetchone()
    if (question_type == "structure"):
        return data[1]
    elif (question_type == "reading"):
        return data[2]
```

```
    elif (question_type == "listening"):
        return data[3]

def createSession(senderPsid, question_type):
    cur, conn = connect()
    query = "SELECT * FROM users WHERE id = %s"
    cur.execute(query, (senderPsid,))
    data = cur.fetchone()

    if (data == None):
        cur, conn = connect()
        query_add = "INSERT INTO `users` (`id`,
`level_structure`, `level_reading`,
`level_listening`) VALUES (%s, '0', '0', '0');"
        cur.execute(query_add, (senderPsid,))
        conn.commit()

    if (checkSession(senderPsid) == 0):
        if (question_type == "structure"):
            if int(str(data[1])) == 0:
                assesment_type = "pre_test"
            else:
                assesment_type = "exercise"
            query_session = "INSERT INTO `sessions`
(`user_id`, `date`, `time`, `question_type`,
`assesment_type`, `status`, `score`) VALUES (%s,
DATE(NOW()), TIME(NOW()), %s, %s, %s, '0');"
            cur.execute(query_session, (senderPsid,
question_type, assesment_type, "processing",))

        elif (question_type == "reading"):
            if int(str(data[2])) == 0:
                assesment_type = "pre_test"
            else:
                assesment_type = "exercise"
            query_session = "INSERT INTO `sessions`
(`user_id`, `date`, `time`, `question_type`,
`assesment_type`, `status`, `score`) VALUES (%s,
DATE(NOW()), TIME(NOW()), %s, %s, %s, '0');"
            cur.execute(query_session, (senderPsid,
question_type, assesment_type, "processing",))
```

```
        elif (question_type == "listening"):
            if int(str(data[3])) == 0:
                assesment_type = "pre_test"

            else:
                assesment_type = "exercise"
            query_session = "INSERT INTO `sessions`
(`user_id`, `date`, `time`, `question_type`,
`assesment_type`, `status`, `score`) VALUES (%s,
DATE(NOW()), TIME(NOW()), %s, %s, %s, '0');"
            cur.execute(query_session, (senderPsid,
question_type, assesment_type, "processing",))
        conn.commit()
        return cur.lastrowid

    else:
        stopConfirmation(senderPsid)
        return 0

def questionStructure(senderPsid):
    question_type = "structure"
    session_id = createSession(senderPsid,
question_type)
    print("SESSION ID : " + str(session_id))
    if (session_id == 0):
        print("Please Stop Last Assesment")

    else:
        cur, conn = connect()
        level = checkLevel(senderPsid,
question_type)
        query_question= "SELECT * FROM
structure_question WHERE level = %s ORDER BY RAND()
LIMIT 5"
        cur.execute(query_question, (level,))
        data_question = cur.fetchall()

        for x in data_question:
            query_session = "INSERT INTO
`questions` (`id`, `session_id`, `question_id`)
VALUES (NULL, %s, %s);"
            cur.execute(query_session,
(str(session_id), str(x[0]),))
            conn.commit()
        sendQuestion(senderPsid, str(session_id),
question_type)
```

```
def questionReading(senderPsid):
    question = []
    question_type = "reading"
    session_id = createSession(senderPsid,
question_type)
    print("SESSION ID : " + str(session_id))

    if (session_id == 0):
        print("Please Stop Last Assesment")
    else:
        cur, conn = connect()
        level = checkLevel(senderPsid, question_type)
        query = "SELECT * FROM reading_material WHERE
reading_material.level = %s ORDER BY RAND()"
        cur.execute(query, (level,))
        data = cur.fetchall()

        for i in data:
            query_question= "SELECT * FROM
reading_question WHERE reading_material_id = %s"
            cur.execute(query_question, (i[0],))
            data_question = cur.fetchall()
            for x in data_question:
                question.append(x)

        for item in islice(question, 5):
            query_session = "INSERT INTO `questions`
(`id`, `session_id`, `question_id`) VALUES (NULL, %s,
%s);"
            cur.execute(query_session,
(str(session_id), str(item[0]),))
            conn.commit()
        sendQuestion(senderPsid, str(session_id),
question_type)

def questionListening(senderPsid):
    question = []
    question_type = "listening"
    session_id = createSession(senderPsid,
question_type)
    print("SESSION ID : " + str(session_id))

    if (session_id == 0):
        print("Please Stop Last Assesment")
```

```
    else:
        cur, conn = connect()
        level = checkLevel(senderPsid, question_type)
        query = "SELECT * FROM listening_material
WHERE listening_material.level = %s ORDER BY RAND()"
        cur.execute(query, (level,))
        data = cur.fetchall()

        for i in data:
            query_question= "SELECT * FROM
listening_question WHERE listening_material_id = %s"
            cur.execute(query_question, (i[0],))
            data_question = cur.fetchall()
            for x in data_question:
                question.append(x)

        for item in islice(question, 5):
            query_session = "INSERT INTO `questions`
(`id`, `session_id`, `question_id`) VALUES (NULL, %s,
%s);"
            cur.execute(query_session,
(str(session_id), str(item[0]),))
            conn.commit()
        sendQuestion(senderPsid, str(session_id),
question_type)

def getPostTest(senderPsid, data):
    cur, conn = connect()
    query_session = "INSERT INTO `sessions`
(`user_id`, `date`, `time`, `question_type`,
`assesment_type`, `status`, `score`) VALUES (%s,
DATE(NOW()), TIME(NOW()), %s, %s, %s, '0');"
    cur.execute(query_session, (senderPsid, data[4],
"post_test", "processing",))
    conn.commit()
    session_id = cur.lastrowid
    level = checkLevel(senderPsid, data[4])

    if (data[4] == "structure"):
        query_question= "SELECT * FROM
structure_question WHERE level = %s ORDER BY RAND()
LIMIT 5"
        cur.execute(query_question, (level,))
        data_question = cur.fetchall()
```

```
        for x in data_question:
            query_session = "INSERT INTO `questions`
(`id`, `session_id`, `question_id`) VALUES (NULL, %s,
%s);"
            cur.execute(query_session,
(str(session_id), str(x[0]),))
            conn.commit()
        sendQuestion(senderPsid, str(session_id),
data[4])

    elif (data[4] == "reading"):
        question = []
        query = "SELECT * FROM reading_material WHERE
reading_material.level = %s ORDER BY RAND()"
        cur.execute(query, (level,))
        reading_material = cur.fetchall()

        for i in reading_material:
            query_question= "SELECT * FROM
reading_question WHERE reading_material_id = %s"
            cur.execute(query_question, (i[0],))
            data_question = cur.fetchall()
            for x in data_question:
                question.append(x)

        for item in islice(question, 5):
            query_session = "INSERT INTO `questions`
(`id`, `session_id`, `question_id`) VALUES (NULL, %s,
%s);"
            cur.execute(query_session,
(str(session_id), str(item[0]),))
            conn.commit()
        sendQuestion(senderPsid, str(session_id),
data[4])

    elif (data[4] == "listening"):
        question = []
        query = "SELECT * FROM listening_material
WHERE listening_material.level = %s ORDER BY RAND()"
        cur.execute(query, (level,))
        listening_material = cur.fetchall()
        for i in listening_material:
            query_question= "SELECT * FROM
listening_question WHERE listening_material_id = %s"
            cur.execute(query_question, (i[0],))
            data_question = cur.fetchall()
```

```
            for x in data_question:
                question.append(x)

        for item in islice(question, 5):
            query_session = "INSERT INTO `questions`
(`id`, `session_id`, `question_id`) VALUES (NULL, %s,
%s);"
            cur.execute(query_session, (str(session_id),
str(item[0]),))
            conn.commit()
        sendQuestion(senderPsid, str(session_id),
data[4])

def handleMessage(senderPsid, receivedMessage):
    print("Received Message : " + str(receivedMessage))
    session = checkSession(senderPsid)

    if (session == 0):
        if (receivedMessage == 'start' or
receivedMessage == 'Start'):
            print("==============START
CHATBOT==============")
            sendCategory(senderPsid)
        elif (receivedMessage == 'level' or
receivedMessage == 'Level'):
            cur, conn = connect()
            query = "SELECT * FROM users WHERE id = %s"
            cur.execute(query, (senderPsid,))
            data = cur.fetchone()

            if (data == None):
                cur, conn = connect()
                query_add = "INSERT INTO `users` (`id`,
`level_structure`, `level_reading`, `level_listening`)
VALUES (%s, '0', '0', '0');"
                cur.execute(query_add, (senderPsid,))
                conn.commit()
            else:
                query_check = "SELECT * FROM users WHERE
id = %s"
                cur.execute(query_check, (senderPsid,))
                current_level = cur.fetchone()
            conn.commit()
            sendMessage(senderPsid, "Level Anda Saat
Ini\n" + "Structure : " + str(current_level[1]) + '\n' +
"Reading : " + str(current_level[2]) + '\n' + "Listening
: " + str(current_level[3]))
```

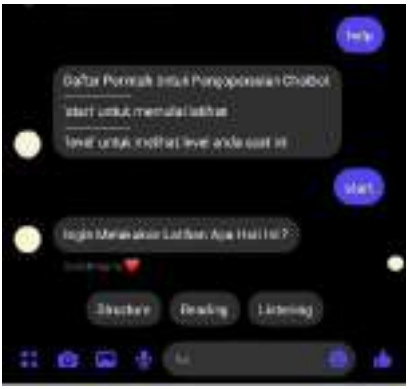
```
        elif (receivedMessage == 'history' or
receivedMessage == 'History'):
            cur, conn = connect()
            query = "SELECT * FROM sessions WHERE
user_id = %s AND status = 'completed' ORDER by `id`
DESC LIMIT 4"
            cur.execute(query, (senderPsid,))
            data = cur.fetchall()
            for x in data:
                sendMessage(senderPsid, "Tanggal : " +
str(x[2]) + '\n' + "Waktu : " + str(x[3]) + '\n' +
"Jenis : " + str(x[4]) + '\n' + "Nilai : " +
str(x[7]))
            conn.commit()
        elif (receivedMessage == 'help' or
receivedMessage == 'Help'):
            sendMessage(senderPsid, "Daftar Perintah
Untuk Pengoperasian Chatbot\n------------------
\n'start' untuk memulai latihan\n------------------
\n'level' untuk melihat level anda saat ini")
        else:
            sendMessage(senderPsid, "Perintah Tidak
Ditemukan, Ketik Help Untuk Menu Bantuan")
    else:
        if 'stop' or 'Stop' in receivedMessage:
            sendMessage(senderPsid, "Sesi Telah
Diakhiri")
            cur, conn = connect()
            session_id = checkSession(senderPsid)
            query_update = "UPDATE `sessions` SET
`status` = 'cancel' WHERE `sessions`.`id` = %s"
            cur.execute(query_update,(session_id,))
            query_delete = "DELETE FROM `questions`
WHERE `questions`.`session_id` = %s"
            cur.execute(query_delete,(session_id,))
            conn.commit()
        elif 'question' or 'Question' in
receivedMessage:
            cur, conn = connect()
            query = "SELECT * FROM sessions WHERE `id`
= %s"
            cur.execute(query, (session,))
            data = cur.fetchone()
            sendQuestion(senderPsid, str(session),
data[4])
        else:
            sendMessage(senderPsid, "Masih ada sesi
yang aktif. kirim 'question' untuk menampilkan soal
kembali, dan 'stop' untuk mengakhiri")
```

```
@app.route("/", methods=['GET', 'POST'])
def webhook():
    if request.method == 'GET':

        if request.args.get("hub.verify_token") ==
VERIFY_TOKEN:
            return request.args.get("hub.challenge")

        else:
            return "Not Connected"

    elif request.method == 'POST':
        data = request.data
        body = json.loads(data.decode('utf-8'))

        if 'object' in body and body['object'] ==
'page':
            entries = body['entry']

            for entry in entries:
                webhookEvent = entry['messaging'][0]
                senderPsid =
webhookEvent['sender']['id']

                if 'message' in webhookEvent:
                    webhookEventMessage =
webhookEvent['message']

                    if 'is_echo' in
webhookEventMessage:
                        return '1'

                    else:
                        global assesment_progress
                        global assesment_type
                        global question_number
                        global post_test_progress
                        global question
                        global assesment_type

                        if 'quick_reply' in
webhookEventMessage:
                            if
webhookEventMessage['quick_reply']['payload'] ==
"structure":
                                print("==============GE
T QUESTION STRUCTURE==============")
                                questionStructure(sende
rPsid)
```

```
                            elif
webhookEventMessage['quick_reply']['payload'] == "reading":
                                print("==============GET
QUESTION READING==============")
                                questionReading(senderPsid)

                            elif
webhookEventMessage['quick_reply']['payload'] ==
"listening":
                                print("==============GET
QUESTION LISTENING==============")
                                questionListening(senderPsid
)

                            elif
webhookEventMessage['quick_reply']['payload'] ==
"post_test_approved":
                                print("==============POST
TEST==============")

                                cur, conn =
connect()
                                query = "SELECT * FROM
sessions WHERE `user_id` = %s AND `status` = 'completed'
ORDER BY `id` DESC LIMIT 1"
                                cur.execute(query,
(senderPsid,))
                                data = cur.fetchone()
                                getPostTest(senderPsid,
data)

                            elif
webhookEventMessage['quick_reply']['payload'] ==
"post_test_cancel":
                                print("==============POST
TEST CANCEL==============")

                                sendMessage(senderPsid,
"Terima Kasih Telah Menggunakan Layanan Kami, Silahkan
ketikan perintah 'start' untuk memulai latihan lagi dan
'help' untuk menu bantuan ")

                            elif
webhookEventMessage['quick_reply']['payload'] ==
"stop_session":
                                print("==============STOP
SESSION==============")

                                cur, conn = connect()
                                session_id =
checkSession(senderPsid)
```

```
                                query_update =
"UPDATE `sessions` SET `status` = 'cancel' WHERE
`sessions`.`id` = %s"
                                cur.execute(query_u
pdate,(session_id,))

                                query_delete =
"DELETE FROM `questions` WHERE
`questions`.`session_id` = %s"
                                cur.execute(query_d
elete,(session_id,))
                                conn.commit()

                            else:
                                saveAnswer(senderPs
id, webhookEventMessage['quick_reply']['payload'])

                        else:
                            handleMessage(senderPsi
d, webhookEventMessage['text'])
                return 'EVENT_RECEIVED', 200
        return '1'
    else:
        return "200"

if __name__ == "__main__":
    app.run()
```

Berikut beberapa hasil yang ditampilkan Chatbot pada Facebook Messenger.



(a)

(b)

(c)

(d)

Gambar 3.11 Hasil respon Chatbot pada Facebook Messenger

BAB IV

PENERAPAN PYTHON UNTUK PENGOLAHAN CITRA DIGITAL

Pada Bab ini akan dijelaskan implementasi pengolahan citra digital untuk pelacakan objek (*Object Tracking*). Pelacakan objek telah banyak digunakan dalam berbagai bidang seperti bidang biomedis, robotic, militer, *game*, dan perfilman. Misalkan pada bidang biomedis pelacakan objek digunakan untuk menentukan status kesehatan hewan yang ditentukan dari gerakannya, pada bidang robotik digunakan untuk pelacakan bola, dan pada bidang militer digunakan untuk pelacakan target pada peluru cerdas.

4.1 Representasi Objek

Objek dapat direpresentasikan dalam berbagai bentuk yang berbeda seperti dalam bentuk titik-titik yang ukurannya sangat kecil dan memiliki karakteristik khusus. Pada frame satu ke frame yang lainnya titik tersebut memiliki karakteristik yang relatif sama. Objek juga dapat direpresentasikan dalam bentuk histogram. Representasi dalam histogram cocok digunakan pada suatu objek yang terdapat perubahan bentuk, ukuran, maupun bentuknya. Yang terkhir ada respresentasi objek dalam kontur. Kontur merupakan merupakan serangakaian piksel terluar dari suatu objek. Representasi ini cocok digunakan pada objek yang berubah bentuk.

4.2 Fitur Objek

Fitur dari suatu objek dapat digunakan untuk melacak objek tersebut. Terdapat berbagai macam fitur yang bisa digunakan diantaranya:

4.2.1 Warna

Pada pelacakan objek, warna merupakan fitur yang banyak digunakan kadrena warna adalah fitur yang paling penting pada sebuah objek. Beberapa penelitian cukup menggunakan warna saja untuk pelacakan objek. Tetap pada implementasinya warna juga memiliki kelemahan ketika objek yang akan dilacak memiliki kemiripan warna dengan latar belakangnya. Kelemahan yang lainnya yaitu masalah pencahayaan menyebabkan warna suatu objek yang tertangkap tidak terlihat seperti warna aslinya.

4.2.2 Tekstur

Ketika warna saja tidak cukup untuk mendapatkan fitur suatu objek dengan kendala warna target yang menyerupai warna objek yang lainnya, tekstur dapat digunakan untuk mengatasi masalah tersebut. Tekstur ini sangat cocok ketika target memiliki yang sama dengan objek yang lainnya tetapi memiliki pola yang berbeda. Contohnya yaitu harimau yang berada di rumput yang kering memiliki warna yang sama. Tetapi harimau memiliki pola kulit dengan garis-garis hitam. Pada pengolahan citra digital, pola tersebut dapat dideskripsikan kedalam fitur tekstur.

4.2.3 Bentuk

Bentuk bisa menjadi pilihan yang tepat pada pelacakan objek ketika objek-objek yang tertangkap pada suatu video memiliki bentuk yang berbeda-beda. Pada proses pelacakan objek bisa menggunakan *template matching* untuk mendeteksi bentuk dasar dengan pencocokan kontur objek. Kelebihan pelacakan objek berdasarkan bentuk yaitu tidak terpengaruh dengan perubahan warna akibat pencahaaan.

4.3 Pemilihan Objek

Pemilihan objek dilakukan untuk mengambil objek target pada suatu video. Terdapat beberapa cara dalam memilik objek diantaranya yaitu:

4.3.1 Membuat persegi yang melingkupi objek

Metode ini merupakan metode sederhana yang umum digunakan. Pada metode ini objek target ditandai dengan menggambarkan persegi yang melingkupi seluruh bagian objek. Kelemahan dari metode ini yaitu latar belakang dari objek yang dipilih masuk kedalam representasi objek yang dibangun. Sehingga masalah ini dapat menyebabkan kegagalan karena pada keadaan tertentu latar belakang tersebut bisa dianggap sebagai objek.

4.3.2 Foreground Extraction

Metode ini menggunakan formula tertentu untuk mengambil objek. Metode ini mengambil bagian tengah objek sebagai representasi objek yang diambil. Bagian pusat objek merupakan bagian yang berikan bobot positif yang tinggi dan bagian luar objek merupakan bagian yang diberikan bobot positif.

4.4 Pendeteksian Objek

Deteksi objek adalah proses yang paling pentin pada pelacakan objek. Inti dari pelacakan objek yaitu pemetaan hasil deteksi objek dari suatu frame dengan hasil frame berikutnya. Pelacakan objek akan bermasalah ketika terdapat kesalahan deteksi pada beberapa frame. Sebelum proses pendeteksian objek, yang harus dilakukan terlebih dahulu yaitu menentukan fitur untuk merepresentasikan objek. Kemudian memilih frame yang memiliki kemiripan dengan objek yang telah direpersentasikan tersebut. Fitur dan representasi objek tersebut sangat mempengaruhi proses pemilihan frame yang memiliki objek yang mirip.

4.5 Pelacakan Objek

Pelacakan objek bertujuan untuk mendapatkan pergerakan objek pada suatu video. Terdapat beberapa metode yang dapat digunakan untuk pelacakan objek. Metode sederhana yang bisa digunakan untuk pelacakan objek yaitu menggunakan fitur warna dengan menggunakan rentang nilai H, S, dan V untuk

mendefinisikan objek. Namun metode tersebut masih kurang andal. Metode lain yang bisa diimplementasikan yaitu KCF.

4.6 Implementasi Python untuk Pelacakan Objek

4.6.1 Pelacakan Objek Menggunakan rentang HSV

Implementasi Python untuk pelacakan objek menggunakan rentang HSV dapat menggunakan program sebagai berikut:

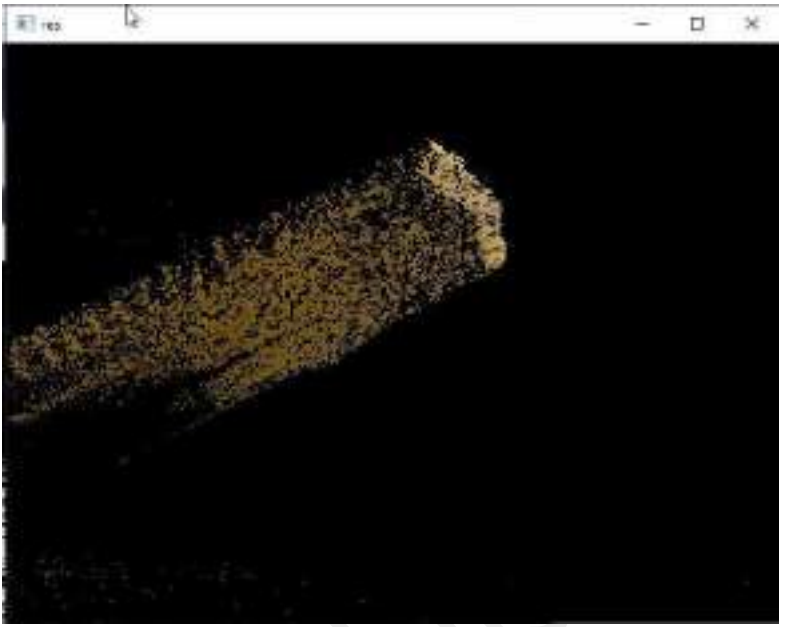
```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)

while(1):

    # Mengambil tiap frame
    _, frame = cap.read()

    # Konfersi BGR ke HSV
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

```
    # mendefinsikan range warna kuning pada HSV
    lower = np.array([20, 100, 100])
    upper = np.array([40, 255, 255])

    # Threshold gambar HSV image untuk mendapatkan
warna kuning
    mask = cv2.inRange(hsv, lower, upper)

    # Bitwise-AND mask dan gambar asli
    res = cv2.bitwise_and(frame,frame, mask= mask)

    cv2.imshow('frame',frame)
    cv2.imshow('mask',mask)
    cv2.imshow('res',res)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cv2.destroyAllWindows()
```

Berikut keluaran dari pelacakan objek menggunakan rentang HSV. Gambar asli ditunjukkan pada Gambar 4.1, gambar hasil threshold dari nilai lower dan upper untuk mendapatkan warna kuning pada suatu frame ditunjukkan pada Gambar 4.2, dan hasil operasi Bitwise-AND mask dengan gambar asli ditunjukkan pada Gambar 4.3.

Gambar 4.1 Hasil tangkapan gambar asli

Gambar 4.2 Hasil threshold dari nilai lower dan upper



Gambar 4.3 Hasil operasi Bitwise-AND mask dengan gambar asli

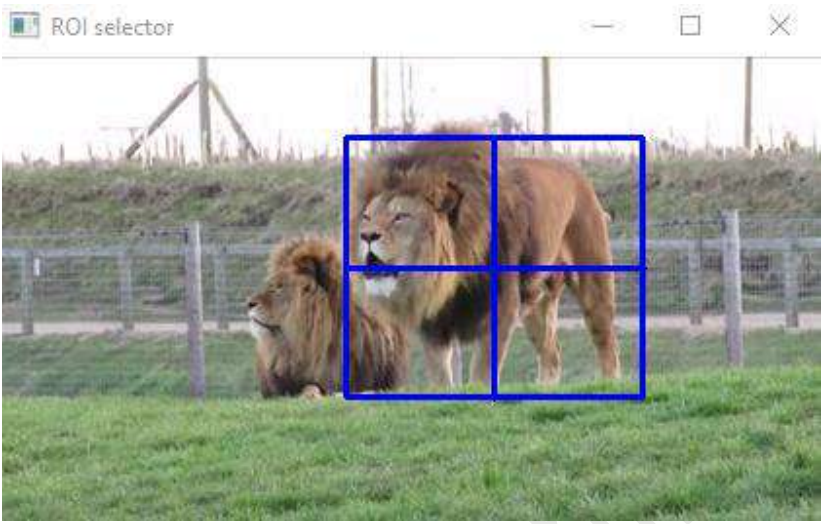
4.6.2 Pelacakan Objek Menggunakan Metode KCF

KCF adalah singkatan dari *Kernelized Correlation Filter* yang merupakan kombinasi dari dua algoritma yaitu Boosting dan MIL Tracker yang berfokus pada perubahan gambar seperti gerakan. Pelacakan objek menggunakan metode KCF dalam OpenCV Python diimplementasikan dalam modul TrackerKCF_create(). Berikut contoh pelacakan objek menggunakan KCF.

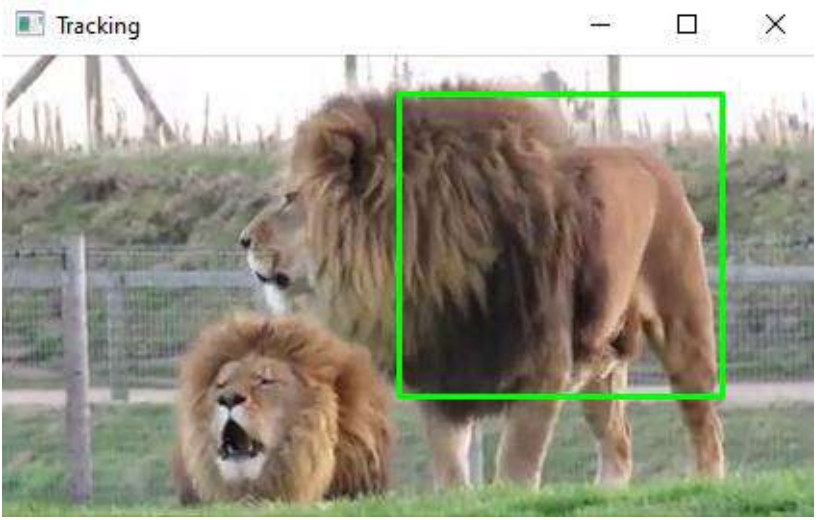
```
import cv2
import numpy as np

# Inisialisasi KCF
tracker = cv2.TrackerKCF_create()
# Inisialisasi video
video = cv2.VideoCapture('video.mp4')

# Mengambil frame pertama video menggunakan fungsi
read()
a,frame=video.read()

# Menampilkan dan menggambar pada frame pertama
menggunakan fungsi selectROI()
# Menyimpan nilainya dalam variabel bbox
bbox = cv2.selectROI(frame)

# Frame dan Posisi (bbox) objek yang dilacak
dinisialisasi menggunakan fungsi init()
a = tracker.init(frame,bbox)

# Insialisasi perulangan while yang melewati frame video
while True:
   # Mengambil frame video dan parameter flag 'a'
menggunakan fungsi
   read()
   # Code ini digunakan untuk memberikan informasi
apakah proses
   pengambilan frame berhasil
   a,frame=video.read()

   # Jika video tidak diambil dengan benar, maka
eksekusi berhenti
   if not a:
        break

   # fungsi update() digunakan untuk memeperbarui frame
disetiap iterasi
   loop
   # Parameter 'a' digunakan untuk menginformasikan
apakah pelacakan
   berhasil atau tidak
   # parameter frame digunakan untuk mengembalikan
posisi objek yang
   dilacak jika parameter pertama benar
   a,bbox=tracker.update(frame)
```

```
   # Jika 'a' benar, blok ini akan dieksekusi.
   # bbox merupakan variabel posisi objek disimpan
   # Fungsi rectangle() pada OpenCV untuk memberikan
bounding box
   disekitar objek pada setiap frame di video
   if a:
        (x,y,w,h)=[int(v) for v in bbox]

cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2,1)

   # Jika tidak dapat melacak ROI yang dipilih code
ini akan memberikan
   "Error" pada frame video
   else:
        cv2.putText(frame,'Error',(100,0),cv2.FONT_HER
SHEY_SIMPLEX,1,(0,0,255),2)

   # Menampilkan frame video pada jendela yang
terpisah
   cv2.imshow('Tracking',frame)

   # Jika pengguna mengklik tombol 'escape', maka
eksekusi berhenti
   if cv2.waitKey(1) & 0XFF== ord('q'):
        break
# destroyAllWindows() digunakan untuk menutup jendela
jika ada yang tersisa
cv2.destroyAllWindows()
```

Keluaran dari frame dari fungsi selectROI() ditampilkan pada Gambar 4.4 dan keluaran pelacakan objek pada video ditampilkan pada Gambar 4.5.



Gambar 4.4 Hasil dari fungsi selectROI()



Gambar 4.5 Hasil pelacakan objek pada video

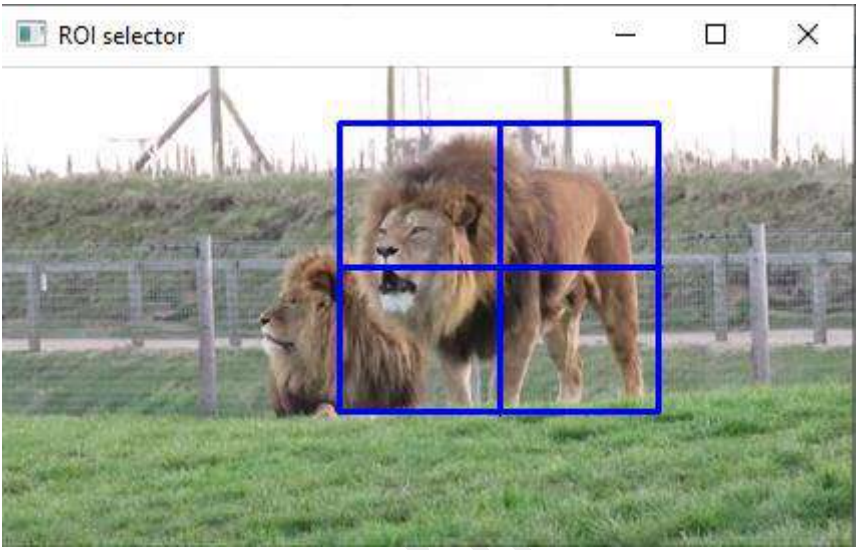
4.6.3 Pelacakan Objek Menggunakan Metode CSRT

CSRT adalah implementasi dari OpenCV dari Channel and Spatial Reliability of Discriminative Correlation Filter (CSR-DCF) merupakan algoritma yang canggih. Pada dasarnya CSR-DCF menggunakan fitur HoG dan spatial reliability maps (SRT) untuk

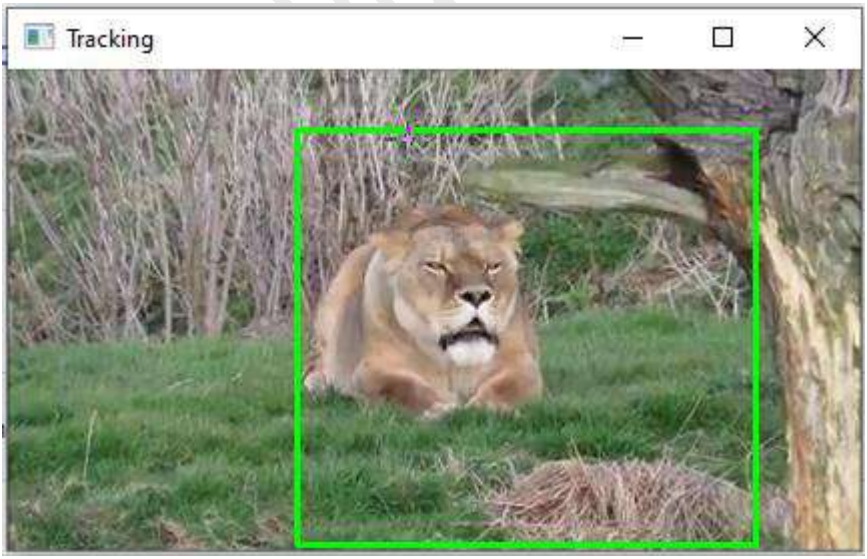
lokalisasi dan pelacakan objek. Pelacakan objek CSRT diimplementasikan dalam modul TrackerCSRT_create() pada Python OpenCV. Program yang digunakan pada metode CSRT sama seperti metode KCF, hanya sedikit mengubah pada nama modul yang digunakan. Berikut program selengkapnya.

```
import cv2
import numpy as np

tracker = cv2.TrackerCSRT_create()
video = cv2.VideoCapture('video.mp4')
a,frame=video.read()
bbox = cv2.selectROI(frame)
a = tracker.init(frame,bbox)

while True:
    a,frame=video.read()
    if not a:
        break
    a,bbox=tracker.update(frame)
    if a:
        (x,y,w,h)=[int(v) for v in bbox]

cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2,1)
    else:
       cv2.putText(frame,'Error',(100,0),cv2.FONT_H
ERSHEY_SIMPLEX,1,(0,0,255),2)

    cv2.imshow('Tracking',frame)

    if cv2.waitKey(1) & 0XFF==27:
        break
cv2.destroyAllWindows()
```

Keluaran dari frame dari fungsi selectROI() ditampilkan pada Gambar 4.6 dan keluaran pelacakan objek pada video ditampilkan pada Gambar 4.7.



Gambar 4.6 Hasil dari fungsi selectROI()



Gambar 4.7 Hasil pelacakan objek pada video

BAB V

PENERAPAN PYTHON UNTUK IMPLEMENTASI LINEAR REGRESSION

5.1 Definisi Linear Regression

Linear regression (Regresi linier) merupakan model statistik yang digunakan untuk memodelkan hubungan antara variabel dependen skalar y dengan variabel satu atau lebih variabel independen yang dilambangkan dengan (x). Variabel terikat merupakan variabel yang akan dijelaskan dan variabel penjelas merupakan variabel yang menjelaskan variabel terikat.

Reperesentasinya berupa persamaan linier yang menggabungkan beberapa nilai input (x) tertentu dan solusinya berupa output hasil prediksi dari kumpulan nilai input (y). Nilai x dan y merupakan nilai numerik. Analisis regresi digunakan untuk memprediksi nilai variabel dependen (y) berdasarkan nilai variabel independen (x) dan menjelaskan dampak perubahan variabel x terhadap variabel y. Regresi linier dapat diasumsikan sebagai berikut:

1. Variabel dependen (y) harus didistribusikan dengan normal dan berkelanjutan.
2. Semua fitur harus independen secara statistik, tidak ada multikolinearitas antar fitur
3. Hubungan yang mendasari antara variabel x dan variabel y yaitu linier.
4. Distribusi probabilitas output y yaitu normal.

5.2 Persamaan Matematika

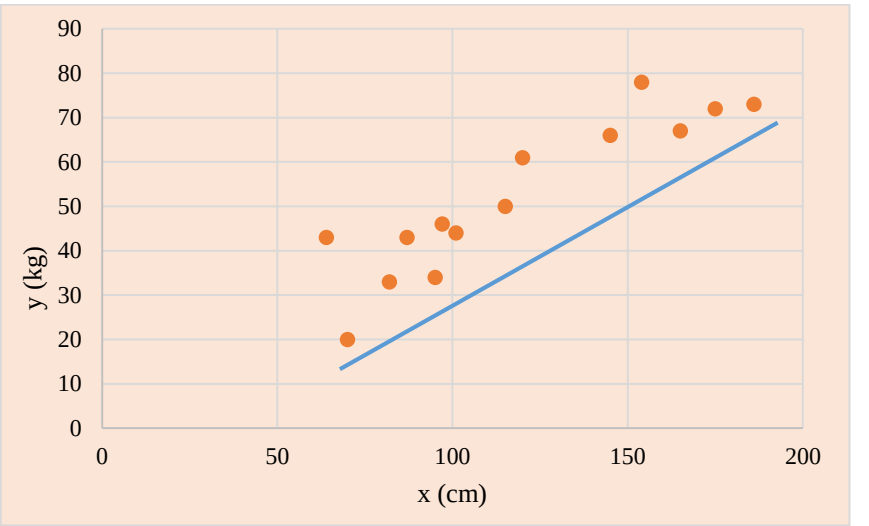
Model regresi liner menyesuaikan fungsi linier ke sekumpulan titik data. Persamaan matematika regeresi linier univariate dapat dituliskan sebagai berikut:

$$y = \beta_0 + \beta_1 x$$

Dimana, $\boldsymbol{\beta_0}$ merupakan nilai awal atau intersep dan $\boldsymbol{\beta_1}$ merupakan slope atau koefisien dari x. Berikut contoh hubungan antara variabel x dan y ditampilkan pada Tabel 4.1 dan Gambar 4.1.

Tabel 5.1 Data tinggi badan dan berat badan untuk variabel x dan y

x (Tinggi Badan)	y (Berat Badan)
70	20
82	33
95	34
64	43
87	43
97	46
120	61
101	44
115	50
186	73
145	66
165	67
175	72
154	78



Gambar 5. 1 Representasi data berdasarkan Tabel 5.1

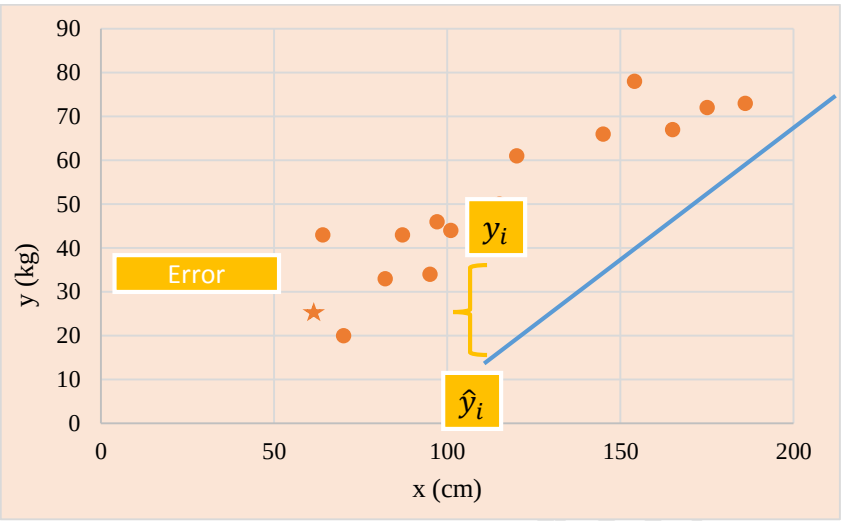
Sedangkan persamaan matematika linear regression multivariate dapat dituliskan sebagai berikut.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_n x_n$$

Persamaan ini mempresentasikan permukaan linier, bidang, dan masih lain-lain. Untuk mempresentasikan garis yang paling sesuai yaitu galat (error) keseluruhan harus sangat rendah.

5.3 Regression Error

Regression error adalah jarak nilai y aktual dengan nilai y prediksi. Contoh regression error dapat dilihat pada Gambar 5.2.



Gambar 5.2 Contoh error regression

Pada regresi, persamaan error merupakan nilai error keseluruhan. Berikut persamaan rata-rata kesalahan (error) kuadrat atau MSE. MSE digunakan sebagai metrik default untuk evaluasi kinerja pada algoritma regresi.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}(y_i - (\beta_0 + \beta_i x))^2$$

Keterangan:

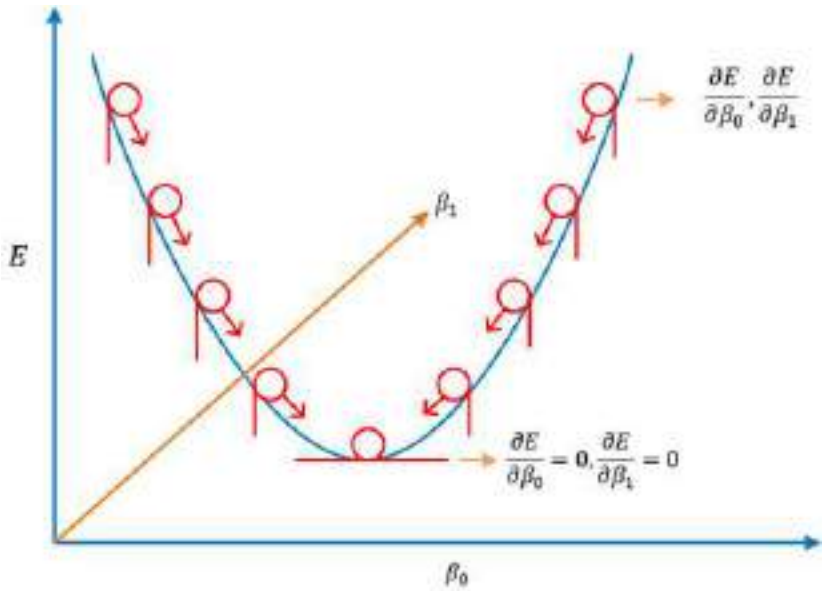
MSE : Mean Squared Error
y_i : Nilai aktual
$\hat{y}_i$: Nilai prediksi

Perintah yang digunakan untuk megetahui nilai MSE pada Python yaitu sebagai berikut.

```
from sklearn.model_selection import mean_squared_error
mean_squared_error (Y_test, yhat)
```

5.4 Optimizer

Optimizer digunakan untuk meminimalisir error. Untuk meminimalisir error keseluruhan dapat menggunakan Gradient Descent Algorithm. Opitimasi dilakukan dengan menyesuaikan nilai β_0 dan β_1. Konsep gradient descent ditampilkan pada Gambar 5.3.



Gambar 5.3 Konsep gradient descent

Keterangan:

$$E = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$Gradien = \frac{\partial E}{\partial \beta_0} = 0, \frac{\partial E}{\partial \beta_1} = 0$$

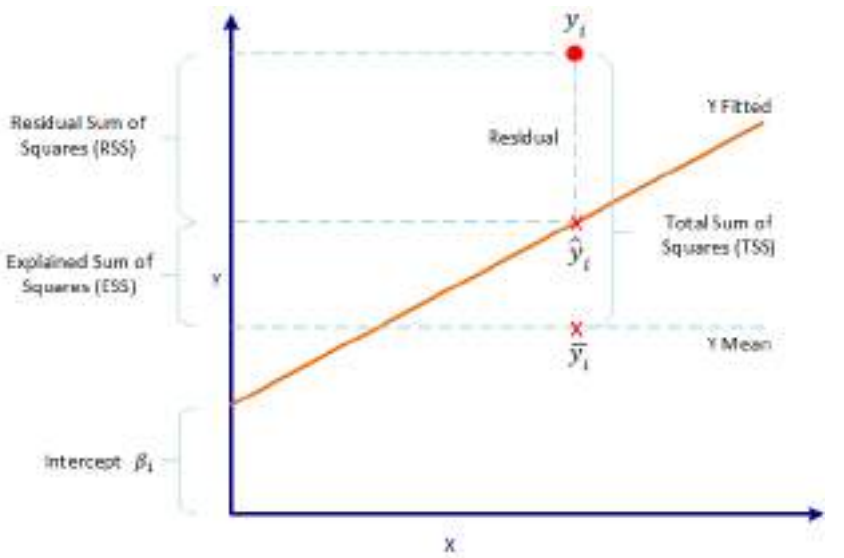
$$\beta_0 baru = \beta_0 lama - \propto \frac{\partial E}{\partial \beta_0}$$

$$\beta_1 baru = \beta_1 lama - \propto \frac{\partial E}{\partial \beta_1}$$

$\propto$ = learning rate (0,0001 – 0,1)

5.5 R-Squared dan Adjusted R-Squared

R-Squared digunakan untuk mengukur akurasi pada regeression atau regresi. R-Squared bernilai 0 – 1. Jika R-Squared semakin mendekati 1, maka model keluaran semakin bagus. Berikut iluastrasi variabel-variabel pada R-squared.



Gambar 4.4 Ilustrasi variabel-variabel pada R-Squared

Rumus R-Squared dapat dituliskan sebagai berikut:

$$R - Squared\ (R^2) = 1 - \frac{RSS}{TSS}$$
$$= \frac{TSS - RSS}{TSS}$$
$$= \frac{ESS}{TSS}$$

Dimana, Explained Sum of Squared (ESS), Residual Sum of Squared (RSS) atau jumlah kuadrat residual, Total Sum of Squared (TSS) atau jumlah kuadrat total dapat dituliskan sebagai berikut.

$$RSS = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

$$TSS = \sum_{i=1}^{N} (y_i - \overline{y_i})^2$$

5.

Berikut perintah yang digunakan untuk mengetahui nilai R-Squared menggunakan Python.

```
from sklearn.model_selection import r2_score
r2_score(Y_test, yhat)
```

5.6 Contoh Implementasi Linear Regression

5.6.1 Import Paket / Library

Import paket atau library Python diantaranya panda, numPy, matplotlib, seaborn.

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

5.6.2 Import Data

Import data yang digunakan untuk prediksi. Data yang digunakan yaitu house_sales_data.csv

```
house_data = pd.read_csv('https://raw.githubusercontent.com/ammishra08/MachineLearning/master/Datasets/house_sales_data.csv', sep = ',')
```

Berikut jika ingin menampilkan data dari house_sales_data.csv.

```
display(house_data)
```

5.6.3 Manipulasi Data

Beberapa manipulasi yang dilakukan diantaranya yaitu:

1. Manipulasi data menggunakan house_data.isnull().sum() digunakan untuk mendeteksi nilai yang hilang berdasarkan kolom. Berikut contohnya:

   ```
   id          0
   date        0
   ```

```
price          0
bedrooms       0
bathrooms      0
sqft_living    0
sqft_lot       0
floors         0
waterfront     0
view           0
condition      0
grade          0
sqft_above     0
sqft_basement  0
yr_built       0
yr_renovated   0
zipcode        0
lat            0
long           0
sqft_living15  0
sqft_lot15     0
dtype: int64
```

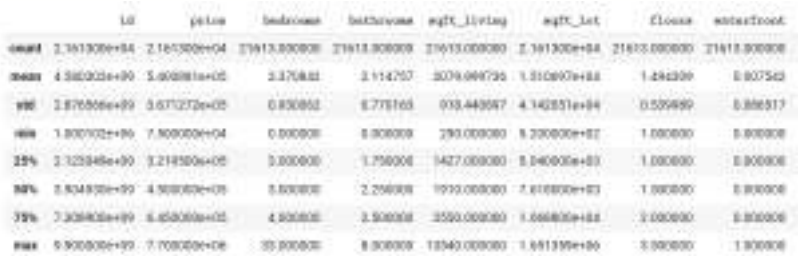
Jika semua kolom bernilai 0, maka tidak ada nilai yang hilang.

2. Kemudian perlu untuk mengetahui informasi data frame yang digunakan seperti tipe data dan memori yang digunakan untuk data yang digunakan tersebut dengan menggunakan perintah house_data.info(). Berikut informasi yang ditampilkan:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            21613 non-null  int64
 1   date          21613 non-null  object
 2   price         21613 non-null  float64
 3   bedrooms      21613 non-null  int64
```

```
 4  bathrooms      21613 non-null  float64
 5  sqft_living    21613 non-null  int64
 6  sqft_lot       21613 non-null  int64
 7  floors         21613 non-null  float64
 8  waterfront     21613 non-null  int64
 9  view           21613 non-null  int64
 10 condition      21613 non-null  int64
 11 grade          21613 non-null  int64
 12 sqft_above     21613 non-null  int64
 13 sqft_basement  21613 non-null  int64
 14 yr_built       21613 non-null  int64
 15 yr_renovated   21613 non-null  int64
 16 zipcode        21613 non-null  int64
 17 lat            21613 non-null  float64
 18 long           21613 non-null  float64
 19 sqft_living15  21613 non-null  int64
 20 sqft_lot15     21613 non-null  int64
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

3. Jika ingin mengetahui informasi data secara numerik dapat menggunakan perintah house_data.describe().



Gambar 4.5 Informasi data numerik menggunakan fungsi describe()

Informasi yang ditampilkan meliputi nilai rata-rata, standart deviasi error dari nilai sample dengan rata-rata, minimal, quartile 1, quartile 2/median, quartile 3, dan maksimal.

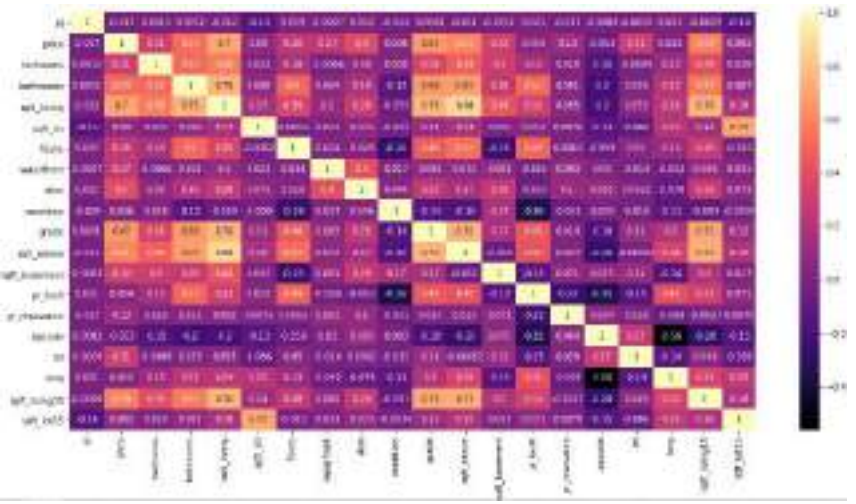
4. Drop data yang tidak perlu seperti id dan tanggal menggunakan house_data.drop(['id', 'date'], axis=1, inplace = True)

5.6.4 Correlation

Correlation secara dasar dapat digunakan pada contoh prediksi harga rumah. Correlation digunakan untuk menampilkan fitur yang penting. Rentang nilai pada correlation mulai dari -1 sampai +1. Berikut perintah yang digunakan untuk menampilkan correlation.

```
plt.figure(figsize=(18, 9))
sns.heatmap(house_data.corr(), annot=True, cmap = 'magma')
```

annot=True digunakan untuk menampilkan data yang sebelumnya tidak ditampilkan menggunakan fungsi drop dan pada correlation ini ingin ditampilkan kembali. Hasil correlation ditampilkan pada Gambar 5.6.



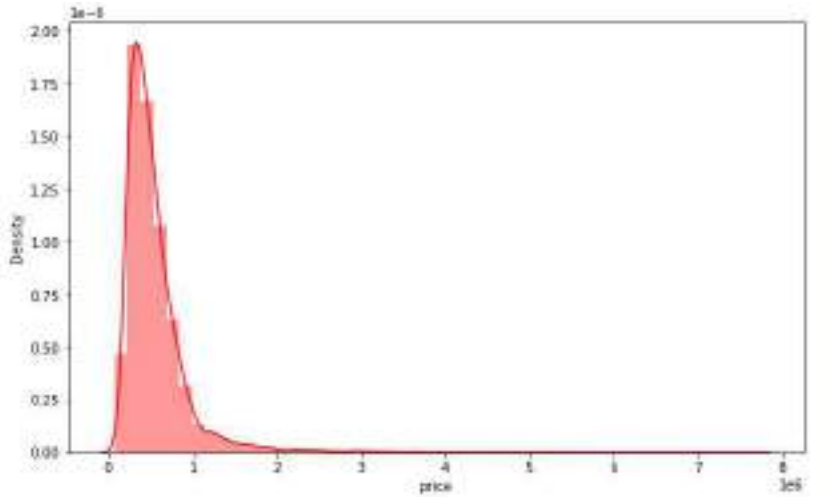
Gambar 5.6 Korelasi antar variable

Semakin tinggi nilai correlation maka keduanya saling mempengaruhi. Misalkan correlation antara bedrooms dengan price bernilai 0,31 yang berarti jumlah kamar tidur terlalu mempengaruhi harga rumah. Dikarenakan nilai correlationnya rendah.
Selanjutnya melihat harga terhadap jumlah harga rumah tersebut dari data melalui grafik menggunakan perintah berikut.

```
plt.figure(figsize=(10,6))
```

```
sns.distplot(house_data['price'], color='r')
```

Hasil grafik ditampilkan pada Gambar 5.7



Gambar 5.7 Grafik harga terhadap jumlah harga rumah

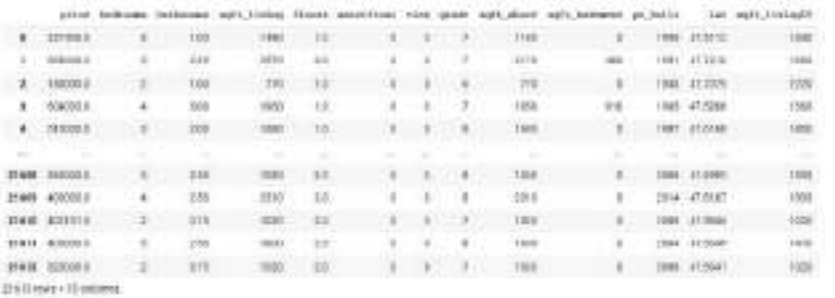
Langkah selanjutnya menampilkan nama kolom dari data frame menggunakan perintah house_data.columns. Berikut hasilnya.

```
Index(['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
'floors','waterfront', 'view', 'condition', 'grade', 'sqft_above',
'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode', 'lat', 'long',
'sqft_living15', 'sqft_lot15'],
dtype='object')
```

Berdasarkan nama kolom diatas, kolom yang tidak dibutuhkan dan memiliki nilai yang sangat rendah pada correlation dapat di drop dan menampilkan kolom yang dibutuhkan.

```
sales = house_data.drop(['sqft_lot', 'condition', 'yr_renovated', 'zipc
ode', 'long', 'sqft_lot15'], axis = 1)
```

Kemudian dilanjutkan dengan run display(sales). Berikut hasil data frame setelah dilakukan drop kolom.



Gambar 5.8 Data frame setelah dilakukan drop beberapa kolom

5.6.5 Split X dan Y

Split X dan Y digunakan untuk memisahkan data yang akan digunakan variabel X dan variabel Y. Berikut perintah inisialisasi data untuk split X dan Y.

```
X = house_df.drop(['price'], axis = 1)
Y = house_df['price']
```

5.6.6 Data Preprocessing

Data preprocessing dimulai dengan import fungsi MinMaxScaler dari paket sklearn.preprocessing untuk memberikan rentang nilai fitur mulai 0 sampai 1. Berikut perintah selengkapnya.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range = (0,1))
```

Paket sklearn.preprocessing menyediakan beberapa fungsi umum dan transformator untuk mengubah vektor fitur mentah menjadi representasi yang lebih sesuai. Untuk menyesuaian penskalaan data yang tersedia dapat dilakukan menggunakn fungsi fit() seperti berikut.

```
scaler.fit(X)
```

Berikut hasilnya

```
MinMaxScaler(copy=True, feature_range=(0, 1))
```

Kemudian untuk menerapkan skala ke data dapat menggunakan fungsi transform()seperti berikut.

```
X_scaler = scaler.transform(X)
```

5.6.7 Split Data Latih dan Uji Menggunakan Cross Validation

Split data menjadi data latih dan data uji, dimulai dengan import fungsi train_test_split dari paket sklearn.preprocessing. Kemudian untuk memisahkan data latih dan data uji selengkapnya dapat dilakukan seperti berikut.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X_scaler, Y, test_si
ze = 0.25, random_state = 42)
```

Keterangan:

- X_train merupakan kelompok data yang digunakan untuk pelatihan
- X_test merupakan kelompok data yang akan digunakan untuk penguian
- Y_train merupakan target kelompok data pelatihan
- Y_test merupakan target kelompok data pengujian
- test_size = 0.25 berarti data yang digunakan untuk pengujian yaitu 25% dari data keseluruhan dan 75% untuk data pelatihan
- random_state = 42 berarti setiap menjalankan kode program untuk memisahkan data latih dan data uji secara acak hasilnya akan sama. Jika random_state = 0, setiap menjalankan kode program nilai acak dari data latih dan uji hasilnya selalu berubah.

5.6.8 Linear Regression

Pada linear regression, pertama yang harus dilakukan yaitu import LinearRegression dari paket sklearn.linear_model. Kemudian inisialisasi linear model seperti berikut.

```
from sklearn.linear_model import LinearRegression

linear_model = LinearRegression(normalize = True)
```

Selanjutnya model dapat dilatih dengan dataset yang telah diatur.
linear_model.fit(X_train, Y_train)

Berikut hasilnya
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=True)

Untuk mengetahui akurasi dapat dituliskan dengan menuliskan perintah berikut.
linear_model.score(X_test, Y_test)

Akurasi yang diperoleh yaitu 0.7010885854735052

5.6.9 Metrics of Regression

Untuk membuat prediksi menggunakan regresi, berikut perintah-perintah yang harus dituliskan.
yh = lr_model.predict(X_test)
display(yh)

Berikut hasilnya.
array([459549.12306492, 751686.09586427, 1235635.86664117, ...,
421131.51084843, 612376.44310812, 439974.35708308])

Untuk menampilkan nilai β_0 dapat menggunakan perintah berikut.
lr_model.intercept_

Nilai β_0 yaitu -445934.61856118636

Untuk menampilkan nilai β_1, β_2, dan seterusnya dapat menggunakan perintah berikut.
lr_model.coef_

Nilai β_1, β_2, dan seterusnya yaitu sebagai berikut.
array([-1.15108094e+06, 3.72243793e+05, -1.87852470e+19,
1.58181050e+05,

2.19876269e+04, 5.58244920e+05, 2.15929396e+05, 9.80298330e+04,
1.13608138e+06, 1.29299210e+19, 6.83357665e+18, -3.09267618e+05,
4.03370709e+04, -1.09190351e+05, 3.71241948e+05, -2.33787679e+05,
1.24138171e+05, -2.87659095e+05])

-1.15108094e+06 merupakan nilai β_1, 3.72243793e+05 merupakan nilai β_2, dan seterusnya.

5.7 Lasso and Ridge Regression

Lasso dan Ridge Regression sangat berkaitan dengan overfitting. Overfitting merupakan suatu kejadian ketika terdapat perbedaan akurasi yang tinggi antara training dan testing. Regression dapat menggunakan regularization (L1 dan L2) untuk menghindari overfitting. Regularization merupakan suatu proses menambahkan informasi. Regularization L1 dan L2 akan ditambahkan sebagai pinalti pada loss function algoritma linear regression untuk menghindari overfitting dan mengkonversi model mejadi sutau model yang umum. Persamaan linear regression tetap sama yaitu dapat dituliskan sebagai berikut:

$$\boldsymbol{y = \beta_0 + \beta_1 x} \quad \text{atau} \quad \boldsymbol{y = x_1 w_1 + x_2 w_2 + x_3 w_3 + b w_n}$$

w_1, w_2, w_3, w_n merupakan weights dan disebut koefisien dari input (x_1, x_2, x_3, dan bias (b)).

5.7.1 Lasso Regression

Lasso atau Least Absolute Shrinkage and Selection Operator Regression merupakan model regresi yang mengimplementasikan Regularization L1. Loss 1 (L1) merupakan nilai absolut dari besaran. Pada lasso regression, nilai absolut dari besar koefisien penalti ditambahkan ke loss function.

$$Loss\ function = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2$$

$$pinalti = \lambda \sum_{j=1}^{p} |\beta_j|$$

$$L1 = \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

p = output yang diprediksi
β_j = koefisien dari nilai loss
λ = besarnya penyusutan
ketika $\lambda = 0$, maka tidak ada parameter yang tereliminasi.
ketika $\lambda = \infty$, maka semakin banyak koefisien yang bernilai 0 dan dihilangkan.

Lasso dapat meyusutkan koefisien fitur yang penting menjadi nol, sehingga menghapus beberapa fitur sekaligus. Hal ini dapat berfungsi dengan baik untuk features selection.

5.7.2 Ridge Regression

Ridge Regression merupakan Model yang mengimplementasikan Regularization L2. Pada Ridge Regression besaran kuadrat dari koefisien sebagai penalti ditambahkan ke loss function.

$$L2 = \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} {\beta_j}^2$$

β_j = koefisien dari input data

5.7.3 Implementasi Menggunakan Python

Untuk mengimplementasikan lasso dan ridge regression menggunakan Python, langkah-langkah yang dilakukan yaitu sebagai berikut.

5.7.3.1 Import Paket yang Digunakan

Untuk memulai lasso dan ridge regression pada Python perlu mengimport paket-paket yang dibutuhkan. Paket-paket yang digunakan diantaranya sebagai berikut.

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

from sklearn.datasets import load_boston
```

5.7.3.2 Memanggil Dataset

Tahapan selanjunya yaitu melakukan inisialisai variabel dan memanggil variabel tersebut untuk import dataset. Dataset yang digunakan yaitu data boston.

```
boston = load_boston()
boston
```

Inisialisasi variabel yang akan digunakan untuk menampilkan data dalam bentuk data frame dapat menggunakan perintah berikut.

```
boston_df = pd.DataFrame(boston.data, columns = boston.feature_
names)
```

boston.data digunakan untuk membaca dataset
boston.feature_names digunakan untuk menambahkan kolom mengguanakan fitur nama

Kemudian dilanjutkan dengan menuliskan perintah boston_df, untuk menampilkan data menjadi data frame. Berikut data dari boston.

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

506 rows × 13 columns

Gambar 5.9 Tampilan data frame boston

Jika ingin menambahkan kolom nilai median harga rumah dalam 1000$ dapat dilakukan dengan menggunakan perintah berikut.

```
boston_df['MEDV'] = boston.target
boston_df
```

Sehingga data frame boston setelah ditambah kolom yaitu sebagai berikut.

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88	11.9

506 rows × 14 columns

Gambar 5.10 Tampilan data frame boston setelah ditambah kolom median

5.7.3.3 Manipulasi Data

Langkah berikutnya yaitu manipulasi data menggunakan isnull untuk mendeteksi nilai yang hilang berdasarkan kolom. berdasarkan kolom pada data frame.

```
boston_df.isnull().sum()
```

Dan berikut hasilnya.

```
CRIM      0
ZN        0
INDUS     0
CHAS      0
NOX       0
RM        0
AGE       0
DIS       0
RAD       0
TAX       0
PTRATIO   0
B         0
LSTAT     0
MEDV      0
dtype: int64
```

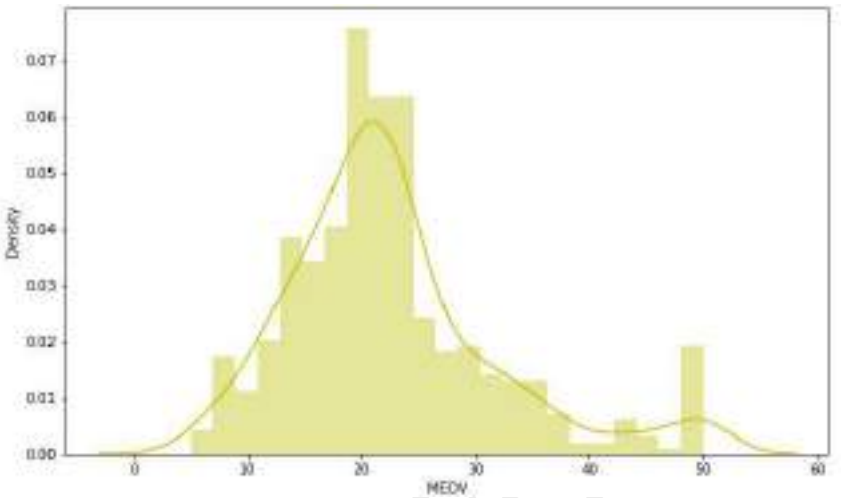
Jika semua kolom bernilai 0. Maka semua kolom dan baris telah memiliki nilai.

5.7.3.4 Asumsi Data

Asumsi digunakan untuk memastikan bahwa data yang digunakan telah sesuai dengan bentuk kurva lonceng sebelum menjalankan regresi. Data akan divisualisasikan dalam bentuk grafik. Berikut perintah untuk asumsi data Median.

```
plt.figure(figsize=(10,6))
sns.distplot(boston_df['MEDV'], color='y')
```

Berikut tampilan grafik asumsi data median.



Gambar 5.11 Asumsi data median

5.7.3.5 Correlation

Correlation digunakan untuk mengetahui hubungan antar variabel. Berikut perintah yang digunakan.

```
plt.figure(figsize=(15,10))
sns.heatmap(boston_df.corr(), annot=True, cmap='magma')
```

Berikut hasil correlation ditampilkan pada Gambar 5.12

Gambar 5.12 Hasil correlation

5.7.3.6 Split X dan Y

Split data X dan Y digunakan untuk memisahkan data yang digunakan untuk variabel X dan Y. Berikut masing-masing data yang digunakan untuk variabel X dan Y.

```
X = boston_df.drop(['MEDV', 'DIS', 'RAD', 'CHAS'], axis = 1)
```

X berisi data keseluruhan kecuali data MEDV, DIS, RAD, dan CHAS

```
Y = boston_df['MEDV']
```

Y berisi data MEDV

5.7.3.7 Data Preprocessing

Data preprocessing menggunakan MinMaxScaler untuk memberikan rentang nilai fitur mulai 0 sampai 1. Berikut beberapa perintah yang dilakukan.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range = (0,1))
scaler.fit(X)
X_scaler = scaler.transform(X)
```

5.7.3.8 Split Data Menjadi Data Train dan Data Test

Langkah berikutnya yaitu membagi dataset menjadi data train dan data test menggunakan perintah berikut.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X_scaler, Y, test_si
ze = 0.2, random_state = 3)
```

5.7.3.9 Lasso Regression

Beberapa perintah untuk lasso regression yaitu sebagai berikut.

a) Import lasso

```
from sklearn.linear_model import Lasso
```

b) Inisialiasi Lasso

```
lasso = Lasso(alpha= 0.01, normalize= True, max_iter=5000)
```

alpha merupakan konstanta yang dikalikan dengan suku L1.
Jika normalize= True, maka regresor X akan dinormalisasi.
max_iter merupakan jumlah maksimal iterasi

c) Fit Lasso Regression Model

```
lasso.fit(X_train, Y_train)
```

Perintah diatas digunakan untuk fit model dengan coordinate descent.

d) Nilai R-Squared Lasso

```
lasso.score(X_test, Y_test)
```

Berikut nilai skor R-Squared
0.7228071413399801

e) Koefisien Model

```
lasso.coef_
```

Berikut nilai koefisien model.

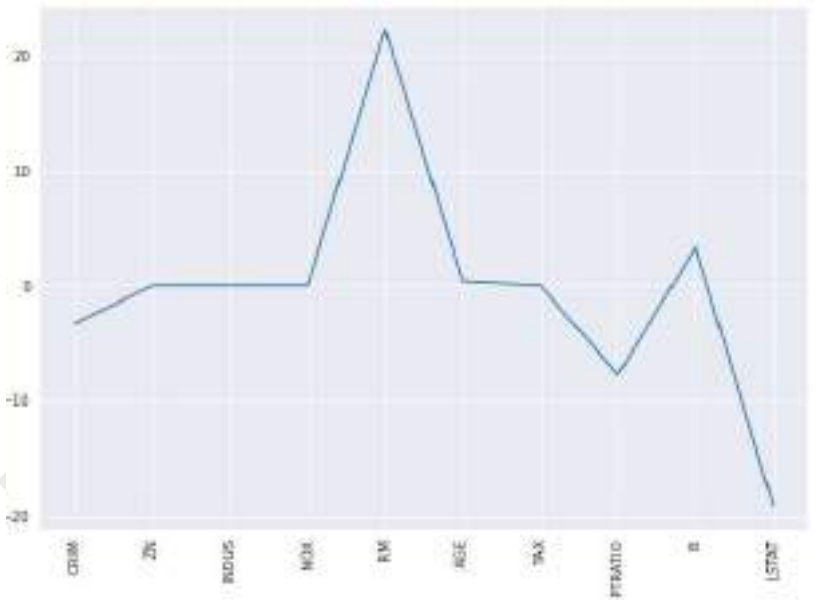
```
array([ -3.28035686,   0.        ,  -0.        ,  -0.        ,
        22.21425436,   0.37784851,   0.        ,  -7.71497034,
```

3.32098007, -19.08885303])

f) Membuat plot koefisien
Plot koefisien digunakan untuk melihat bagaimana nilai alpha memengaruhi koefisien lasso.

```
sns.set_style('darkgrid')
plt.figure(figsize=(10,7))
plt.plot(range(len(X.columns)), lasso.coef_)
plt.xticks(range(len(X.columns)), X.columns, rotation = 90)
plt.show()
```

Berikut hasil plot koefisien lasso.



Gambar 5.13 Plot koefisien lasso regression

5.7.3.10 Ridge Regression

Beberapa perintah untuk ridge regression yaitu sebagai berikut.

a) Import Ridge

```
from sklearn.linear_model import Ridge
```

b) Inisialiasi Ridge
Ridge = Ridge (alpha= 0.01, normalize= True, max_iter=5000)
alpha merupakan konstanta yang dikalikan dengan suku L2.
Jika normalize= True, maka regresor X akan dinormalisasi.
max_iter merupakan jumlah maksimal iterasi

c) Fit Ridge Regression Model
Ridge.fit(X_train, Y_train)
Perintah diatas digunakan untuk fit model dengan coordinate descent.

d) Nilai R-Squared Ridge
ridge.score(X_test, Y_test)

Berikut nilai skor R-Squared
0.7179918159913167

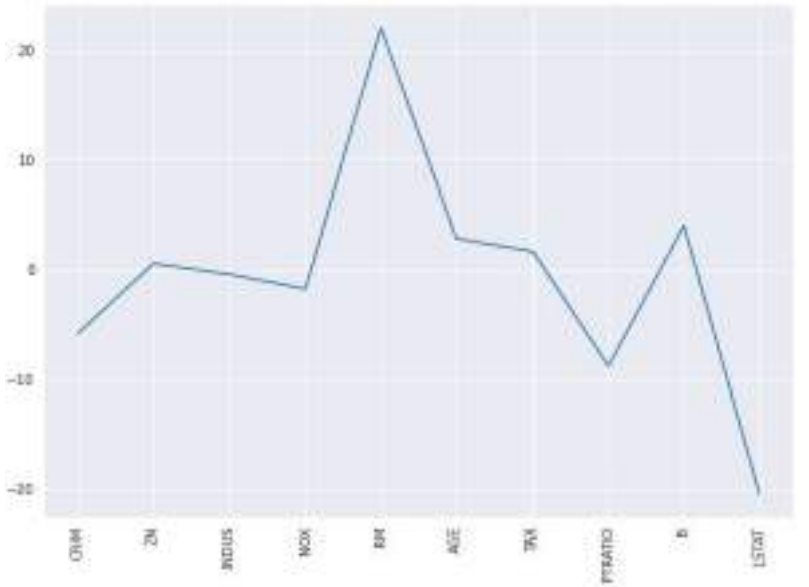
e) Koefisien Model
ridge.coef_

Berikut nilai koefisien model.
array([-5.82632687, 0.55691173, -0.40428325, -1.76865803,
22.00828173, 2.83803875, 1.67454473, -8.82840922,
4.03015492, -20.37043501])

f) Membuat plot koefisien
Plot koefisien digunakan untuk melihat bagaimana nilai alpha memengaruhi koefisien ridge.

```
sns.set_style('darkgrid')
plt.figure(figsize=(10,7))
plt.plot(range(len(X.columns)), ridge.coef_)
plt.xticks(range(len(X.columns)), X.columns, rotation = 90)
plt.show()
```

Berikut hasil plot koefisien ridge.



Gambar 5.14 Plot koefisien ridge regression

BAB VI

DECISION TREE DAN RANDOM FOREST

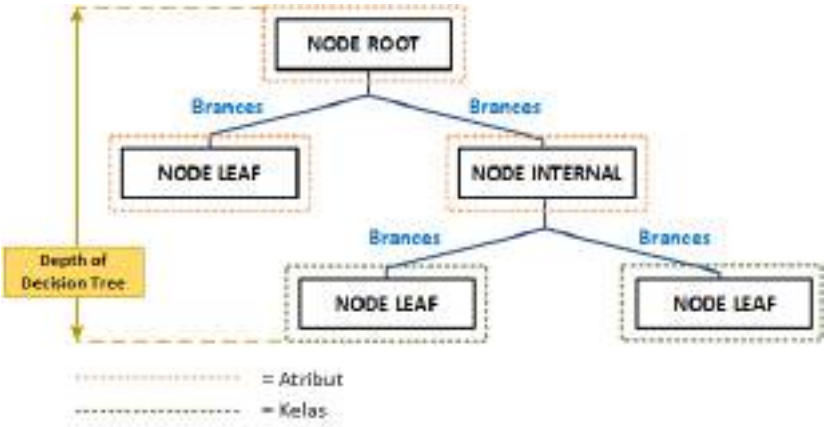
6.1 Decision Tree

Decision Tree merupakan model machine learning yang digunakan untuk klasifikasi dan regresi yang memiliki struktur seperti pohon. Dalam membuat analisi keputusan, decision tree dapat digunakan untuk merepresentasikan keputusan secara visual. Pada machine learning, decision tree digunakan untuk strategi memperoleh tujuan tertentu. Tujuan decision tree yaitu untuk membuat model yang memprediksi nilai variabel target dengan mempelajari beberapa aturan tertentu yang disimpulkan dari fitur data. Kelebihan menggunakan decision tree yaitu:

1. Representasi yang sederhana sehingga mudah untuk dipahami
2. Fleksibel untuk mengubah decision tree agar seimbang dan lebih praktis
3. Algoritma multivariate yang direpresentasikan dalam bentuk diagram sehingga mudah dipahami semua orang
4. Dapat diimplementasikan pada data kategorikal dan kontinu

6.1.1 Ilustrasi

Decision tree dapat mengklasifikasikan suatu contoh atau kasus yang dimulai dari akar pohon dan bergerak sampai simpul daun. Decision tree terdiri dari tiga jenis node yaitu node root, node internal, dan node leaf seperti yang ditampilkan pada Gambar 6.1.



Gambar 6.1 Struktur Decision Tree

Pada decision tree pada setiap node merepresentasikan atribut, setiap brances atau cabang merepresentasikan nilai dari atribut dan setiap node leaf atau simpul daun merepresentasikan kelas. Untuk membangun model decision tree dapat menggunakan algoritma Iterative Dychotomizer Version 3 (ID3). ID3 merupakan algoritma yang dapat membangun decision tree dari atas ke bawah atau dari akar. Mulai dari dari atribut mana yang akan digunakan sebagai root, menjawab pertanyaan dengan mengevaluasi atribut mengggunakan statistik information gain. ID3 menggunakan entropy dan information gain untuk membangun decision tree.

6.1.2 Information Gain dan Entropy

6.1.2.1 Entropy

Entropy digunakan untuk mengukur informasi dari atribut. Berikut rumus persamaan untuk mencari nilai entropy.

$$E(S) = -P_+ log_2 P_+ - P_- log_2 P_-$$

Dimana:

S merupakan data sampel yang digunakan untuk training

E(S) merupakan jumlah kelas positif (+) maupun kelas negatif (-) dari jumlah data acak dari sampel S.

P_+ merupakan jumlah data positif pada sampel suatu atribut.

P_- merupakan jumlah data negatif pada sampel suatu atribut.

6.1.2.2 Information Gain

Information Gain digunakan untuk mengukur seberapa baik atribut memisahkan contoh training berdasarkan klasifikasi tragetnya. Ukuran ini digunakan untuk memilih antribut kandidat saat membangun pohon. Untuk mencari persamaan information gain atau dapat dinotasikan gain yaitu sebagai berikut.

$$Gain(S, A) = E(S) - E(S, A)$$

$$Gain(S, A) = E(S) - \sum_{v \epsilon Value(A)} \frac{|S_v|}{|S|} E(S_v)$$

Dimana:
A merupakan atribut
V merupakan nilai yang mungkin untuk atribut A
$Value(A)$ merupakan himpunan dari semua kemungkinan nilai untuk atribut A
S_v menyatakan jumlah sampel untuk nilai V
S menyatakan seluruh sampel
$E(S_v)$ menyatakan entropy untuk sampel yang memiliki nilai V
$E(S, A)$ menyatakan nilai entropy setelah split

6.1.3 Contoh Decision Tree

Berikut data training yang akan diklasifikasikan menggunakan decision tree yang ditampilkan pada Tabel 6.1.

Tabel 6.1 Data training yang akan diklasifikasikan

Outlook	Suhu	Kelembaban	Berangin	Kelas
Cerah	88	93	Ya	Tidak Bekerja
Cerah	57	70	Tidak	Bekerja
Cerah	77	73	Ya	Bekerja
Cerah	64	91	Ya	Tidak Bekerja
Mendung	80	87	Tidak	Bekerja
Mendung	67	74	Ya	Bekerja
Mendung	90	94	Tidak	Bekerja
Hujan	80	62	Tidak	Bekerja
Hujan	72	69	Tidak	Tidak Bekerja
Hujan	66	64	Ya	Tidak Bekerja

$$E(S) = -\frac{6}{10} log_2 \frac{6}{10} - \frac{4}{10} log_2 \frac{4}{10}$$

$$E(S,A) = \frac{4}{10} \times E_{cerah} + \frac{3}{10} \times E_{mendung} + \frac{3}{10} \times E_{hujan}$$

$$Gain\ (S,A) = E(S) - E(S,A)$$

Dari data tabel diatas akan diubah menjadi model decision tree seperti Gambar berikut.

Gambar 6.2 Contoh Klasifikasi Decision Tree

6.1.4 Kelebihan dan Kekurangan Decision Tree

6.1.4.1 Kelebihan

Kelebihan decision tree diantaranya:

1. Menghasilkan aturan yang mudah dimengerti
2. Melakukan klasifikasi tanpa banyak perhitungan
3. Dapat menangani variabel kontinu maupun kategroi
4. Memberikan indikasi yang jelas untuk prediksi atau klasifikasi

6.1.4.2 Kekurangan

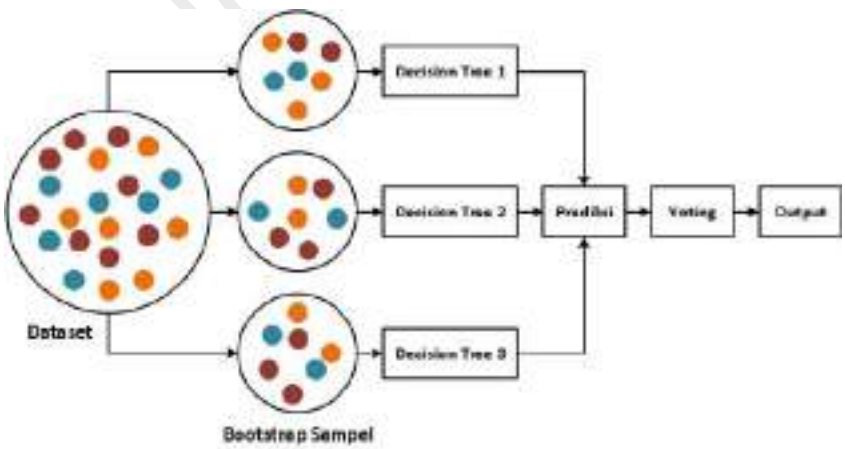
Kekurangan decision tree diantaranya:

1. Tidak cocok untuk prediksi atribut yang kontinu
2. Kinerja yang kurang baik jika terdapat banyak kelas dan data kecil
3. Sulitnya merancang decision tree yang optimal.

6.2 Random Forest

Random forest merupakan algoritma machine learning yang memanfaatkan decision tree untuk membuat keputusan. Random forest menggunaakan metode boostrap untuk membangun decision tree. Pengambilan sampel boostrap merupakan suatu metode yang melibatkan sampel data secara acak dari sampel yang diberikan dan mengganti sumber data.

Proses Boostrap aggregating atau bagging dimulai dengan membagi data pembelajaran menjadi beberapa sampel menggunakan pengambilan sampel bootstrap dan menghasilkan sampel bootstrap dalam jumlah yang besar. Untuk membuat decision tree, setiap sampel Bootstrap dan semua pohon memberikan suara untuk menghasilkan keputusan dengan mempertimbangkan suara terbanyak. Untuk menghasilkan output dari random forest untuk klassifikasi menggunakan voting dari seluruh hasil decision tree. Untuk menghasilkan output dari random forest untuk regresi menggunakan nilai rata-rata dari hasil keseluruhan. Random forest dapat membuat tiga decision tree yang mengambil input dari subset menggunakan bagging seperti yang ditunjukkan di bawah ini:



Gambar 6.3 Random forest menggunakan bagging untuk klasfikasi

6.3 Implementasi Decision Tree dan Random Forest pada Python

Tahapan yang dilakukan untuk implementasi Decision Tree dan Random Forest pada Python diantaranya:

6.3.1 Import Paket Python

Pertama yang harus dilakukan yaitu import paket-paket yang dibutuhkan seperti Panda, numPy, matplotlib.pyplot, dan seaborn.

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

6.3.2 Import Dataset dan Eksplorasi Data

Import dataset yang akan digunakan untuk klasifikasi kemudian eksplorasi dataset tersebut seperti menampilkan data dan informasi data yang telah diimport.

6.3.2.1 Import Dataset

```
churn_df = pd.read_csv('https://raw.githubusercontent.com/ammishra08/MachineLearning/master/Datasets/churn-bigml-20.csv')
```

6.3.2.2 Menampilkan beberapa baris pertama dataset menggunakan fungsi head()

```
churn_df.head()
```

Berikut 5 baris pertama dari dataset.

6.3.2.3 Mendeteksi nilai yang hilang

Mendeteksi nilai yang hilang berdasarkan kolom data frame menggunakan fungsi isnull().

```
churn_df.isnull().sum()
```

Berikut hasilnya.

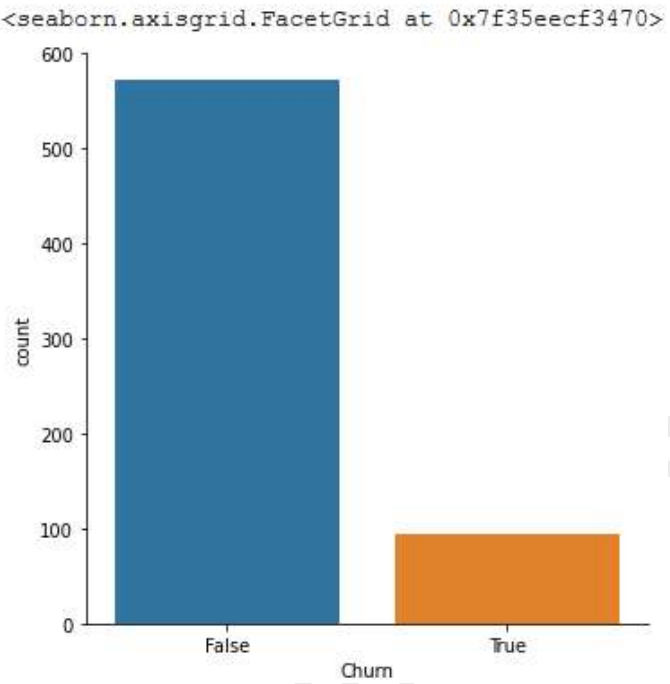
```
State                   0
Account length          0
Area code               0
International plan      0
Voice mail plan         0
Number vmail messages   0
Total day minutes       0
Total day calls         0
Total day charge        0
Total eve minutes       0
Total eve calls         0
Total eve charge        0
Total night minutes     0
Total night calls       0
Total night charge      0
Total intl minutes      0
Total intl calls        0
Total intl charge       0
Customer service calls  0
Churn                   0
dtype: int64
```

6.3.2.4 Visualisasi Data

Menampilkan visualisasi data menggunakan fungsi catplot(). Catplot merupakan teknik visualisasi data menggunakan variabel numerik dan satu atau lebih variabel kategori.

```
sns.catplot(x = 'Churn', kind = 'count', data = churn_df)
```

Berikut hasil fungsi catplot().



Gambar 6.4 Visualisasi data menggunakan fungsi catplot()

6.3.2.5 Menampilkan informasi data frame

Menampilkan informasi data frame seperti jumlah kolom, nama kolom, tipe data, dll menggunakan fungsi info().

```
churn_df.info()
```

6.3.3 Data Preprocessing

6.3.3.1 Menghapus data yang sama

```
churn_df['International plan'].drop_duplicates()
```

drop_duplicates() digunakan untuk menghapus baris yang memiliki data yang sama pada kolom 'International plan'.

6.3.3.2 Import LabelEncoder dan inisialisasi variabelnya

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

LabelEncoder digunakan untuk menormalisasi label sehingga kelas hanya berisi nilai antara 0 dan 1.

6.3.3.3 Fit label encoder

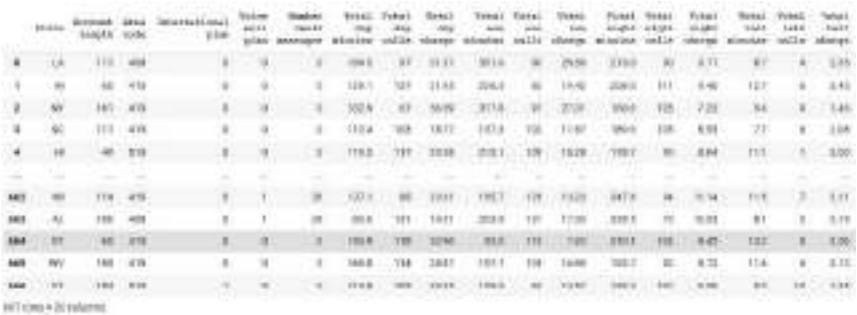
le.fit(churn_df['International plan'].drop_duplicates())
le.fit(churn_df['Voice mail plan'].drop_duplicates())
Perintah diatas digunakan untuk fit label encoder dari 'International plan' dan 'Voice mail plan'.

6.3.3.4 Mengubah label menjadi pengkodean

churn_df['International plan'] = le.transform(churn_df['International plan'])
churn_df['Voice mail plan'] = le.transform(churn_df['Voice mail plan'])
transform() digunakan untuk mengubah label menjadi pengkodean yang dinormalisasi.

6.3.3.5 Menampilkan Churn data frame

churn_df

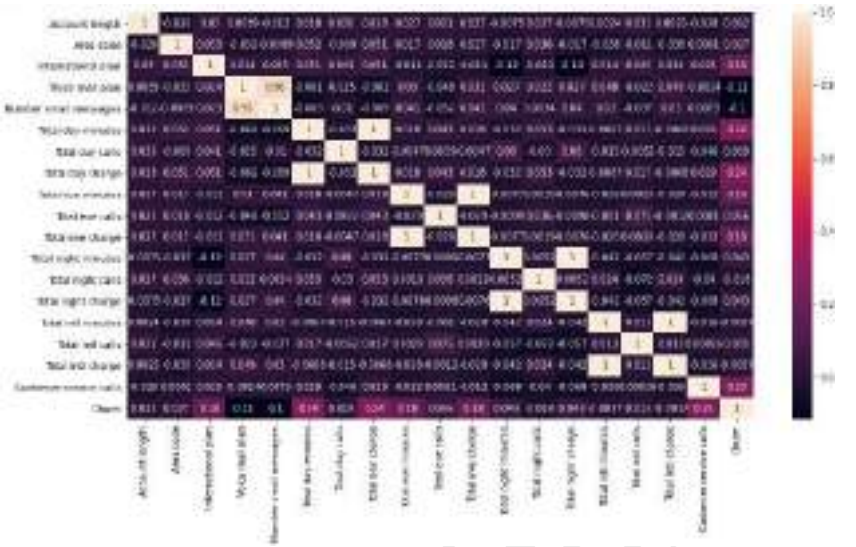


Gambar 6.5 Data frame churn setelah proses preprocessing

6.3.4 Correlation

Menampilkan correlation dari data frame churn menggunakan perintah berikut menggunakan churn_df.corr(). Berikut perintah selengkapnya.

plt.figure(figsize = (15,8))
sns.heatmap(churn_df.corr(), annot=True)
Berikut hasil plot untuk menampilkan correlation.



Gambar 6.6 Korelasi antar variable

6.3.5 Feature Selection

6.3.5.1 Menampilkan nama kolom

Menampilkan nama koom menggunakan perintah berikut.

```
churn_df.columns
```

Berikut nama kolom pada data frame churn.

```
Index(['State', 'Account length', 'Area code', 'International plan',
       'Voice mail plan', 'Number vmail messages', 'Total day minutes',
       'Total day calls', 'Total day charge', 'Total eve minutes',
       'Total eve calls', 'Total eve charge', 'Total night minutes',
       'Total night calls', 'Total night charge', 'Total intl minutes',
       'Total intl calls', 'Total intl charge', 'Customer service calls',
       'Churn'],
      dtype='object')
```

6.3.5.2 Split X dan Y

Berikut nilai dari X dan Y.

```
X = churn_df.drop(['State', 'Churn', 'Number vmail messages', 'Tota
l day charge', 'Total eve charge', 'Total night charge', 'Total intl char
ge'], axis = 1)
```

```
Y = churn_df['Churn']
```

6.3.6 Cross Validation

Import train_test_split kemudian split data menjadi data train dan data test dimana data test sebesar 0,25 atau 25% dari jumlah keseluruhan data.

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state = 2)
```

6.3.7 Klasifikasi Decision Tree

6.3.7.1 Import DecisionTreeClassifier dan inisialisasi variabelnya

Membangun model decision tree menggunakan scikit-learn.

```
from sklearn.tree import DecisionTreeClassifier
clf_tree = DecisionTreeClassifier(criterion = 'entropy', max_depth = 9, random_state = 3)
```

DecisionTreeClassifier digunakan untuk mengklasifikasi dataset pada kelas jamak. Pada DecisionTreeClassifier terdapat beberapa parameter diantaranya criterion = 'entropy' yang digunakan untuk mengukur kualitas split dengan menggunakan pengukuran entropy. Parameter max_depth = 9 digunakan untuk mengatur kedalaman decision tree. Parameter random_state = 3 digunakan untuk mengotrol keacakan dari estimator.

6.3.7.2 Fit data train

```
clf_tree.fit(X_train, Y_train)
```

Perintah diatas digunakan untuk menyesuaikan model dengan dengan data training yang diberikan.

6.3.7.3 Skor akurasi data train dan data test

Mengukur akurasi dari model terhadap data train.

```
clf_tree.score(X_train, Y_train)
```

Hasil akurasi model terhadap data train yaitu 0.992.

Mengukur akurasi dari model terhadap data test.
clf_tree.score(X_test, Y_test)
Hasil akurasi model terhadap data test yaitu 0.8922155688622755.

6.3.8 Plot Decision Tree

6.3.8.1 Import Paket

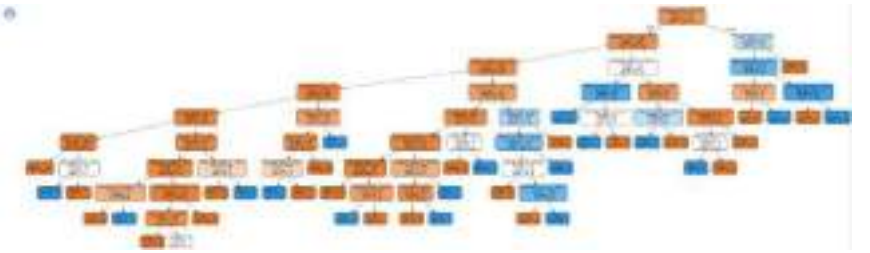
from sklearn.externals.six import StringIO
StringIO menyediakan cara mudah untuk bekerja dengan teks di memori.
from IPython.display import Image
Perintah diatas digunakan untuk memuat dan menampilkan gambar lokal di notebook google colab.
from sklearn.tree import export_graphviz
export_graphviz digunakan untuk mengubah klasifikasi decision tree menjadi dot file.
import pydotplus
pydotplus digunakan untuk mengubah dot file menjadi png atau yang dapat ditampilkan di google colab.

6.3.8.2 Mengubah hasil klasifikasi decision tree menjadi

dot_data = StringIO()
export_graphviz(clf_tree, out_file=dot_data, feature_names=list(X.columns), filled=True, rounded = True)
Perintah diatas digunakan untuk mengubah data klasifikasi decision tree menjadi dot file.
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
export_graphviz
Perintah diatas digunakan untuk mengubah dot file menjadi gambar grafik.

6.3.8.3 Menampilkan Grafik

Image(graph.create_png())



Gambar 6.7 Tampilan hasil klasifikasi decision tree

6.3.9 Matrik Klasifikasi

6.3.9.1 Prediksi data test

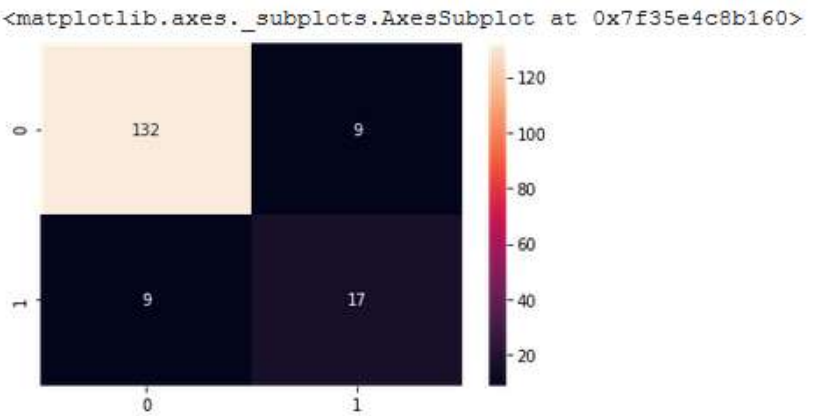
Memprediksi data test menggunakan fungsi predict()
yhat = clf_tree.predict(X_test)

6.3.9.2 Confusion Matriks

Perhitungan confusion matrix menggunakan Python dengan import classification_report, confusion_matrix
from sklearn.metrics import confusion_matrix, classification_report
confusion_matrix(Y_test, yhat)
Perintah diatas digunakan untuk mengevaluasi akurasi dari klasifikasi.

6.3.9.3 Plot data

Membuat plot data yang berbentuk persegi sebagai matriks dengan kode warna.
sns.heatmap(confusion_matrix(Y_test, yhat), annot= True, fmt='0.0f')
Berikut tampilan plot data yang berbentuk persegi.



Gambar 6.8 Tampilan plot data persegi

6.3.9.4 Menampilkan classification report

```
print(classification_report(Y_test, yhat))
```

Berikut hasil klasifikasi.

	precision	recall	f1-score	support
False	0.94	0.94	0.94	141
True	0.65	0.65	0.65	26
accuracy			0.89	167
macro avg	0.80	0.80	0.80	167
weighted avg	0.89	0.89	0.89	167

6.3.10 Random Forest

6.3.10.1 Import model

```
from sklearn.ensemble import RandomForestClassifier
```

6.3.10.2 Klasifikasi Random Forest

```
clf_rf = RandomForestClassifier(bootstrap=True, max_depth=9, n_e
stimators=200, random_state=3, criterion='gini')
```

RandomForestClassifier digunakan untuk pengukur meta yang menggunakan klasifikasi decision tree pada berbagai sub-sampel dari kumpulan data dan menggunakan rata-rata untuk meningkatkan akurasi prediksi dan kontrol over-fitting. Pada RandomForestClassifier terdapat beberapa parameter diantaranya criterion = 'gini' yang digunakan untuk mengukur kualitas split dengan menggunakan pengukuran Gini. Parameter max_depth = 9 digunakan untuk mengatur kedalaman decision tree. Parameter random_state = 3 digunakan untuk mengotrol keacakan dari estimator.

6.3.10.3 Fit data training

clf_rf.fit(X_train, Y_train)
Perintah diatas digunakan untuk menyesuaikan model dengan dengan data training yang diberikan.

6.3.10.4 Membuat prediksi

predictions = clf_rf.predict(X_test)
predict() digunakan untuk membuat prediksi pada data baru kemudian menampilkan hasilnya dengan memanggil predictions. Berikut hasil prediksi.

```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False,  True,  True, False, False, False, False, False,
       False,  True, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
```

False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, True, False, False, False,
False, False, False, False, False])

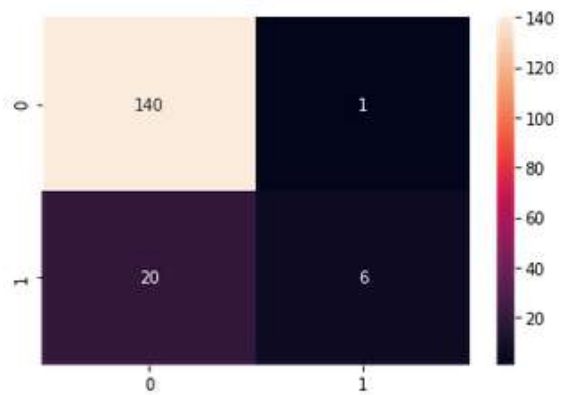
6.3.10.5 Plot data

Membuat plot data yang berbentuk persegi sebagai matriks dengan kode warna.

```
sns.heatmap(confusion_matrix(Y_test, predictions), annot= True, f
mt='0.0f')
```

Berikut tampilan plot data yang berbentuk persegi.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f35e4aefba8>
```



Gambar 6.9 Tampilan plot data persegi

GLOSSARY

Istilah	Keterangan
>>>	Prompt default shell interaktif Python
…	Prompt default shell interaktif Python saat memasukkan kode di bawah blok indentasi atau dalam sepasang pembatas yang cocok. Pembatas dapat berupa kurung, kurung kurawal, atau kurung siku. Tanda ini juga disebut objek elipsis.
argument	Suatu informasi yang diperlukan oleh suatu fungsi agar dapat melakukan tugasnya
bug	Kode yang menyebabkan program gagal dieksekusi dengan benar
BDFL	Benevolent Dictator For Life. Juga dikenal sebagai Guido van Rossum kreator dari Phyton
bytecode	Source code pada Phyton dikompilasi menjadi bytecode yang merupakan representasi internal program Phyton pada Cphyton interpreter
casting	Proses engubah satu tipe data menjadi tipe data yang lain. Contoh angka (number) dapat disimpan sebagai text tetapi perlu dikonversi menjadi integer (misal int(“3”)
commenting	Teks dalam program yang ditujukan untuk kepentingan user dan akan diabaikan oleh program perlu dituliskan sebagai komentar. Untuk menjadikan teks tersebut menjadi komentar maka dimulai dengan simbol hash #

comparative operator	Disebut juga operator logika. Digunakan untuk membandingkan 2 data dalam program. Termasuk di sini adalah (==, <, >, <=.>=)
constant	Bilangan yang bersifat tetap (tidak berubah). Pemberian nama konstanta sebaiknya menggunakan huruf kapital
data-type	Merpakam ragam jenis tipe data yang disimpan di komputer. Misalnya float, integer, dan string
default	Nilai yang diberikan oleh argumen atau variabel sebagai nilai awal (starting point)
equal operator	Tanda sama dengan (=), digunakan untuk menetapkan suatu nilai ke dalam variabel. Misalnya n=2 artinya memberikan nilai 2 pada variabel n
escape sequence	Kombinasi (pasangan) karakter yang dianggap sebagai karakter tunggal karena karakter pertama tidak menampilkan dirinya sendiri (escape) ketika digunakan dalam literal string. Misalnya \n (berarti newline. Tanda \ tidak dieksekusi oleh program)
execute	Arti lain dari execute adalah **run**. Mengeksekusi sejumlah kode untuk menjalankan perintah
float	Tipe data bilangan yang dapat memiliki nilai desimal
function	Potongan kode yang dapat digunakan kembali
global variable	Variabel yang dapat digunakan di bagian manapun dari program
hacking	Mengambil beberapa kode yang ditulis sebelumnya dan menulis ulang untuk membuatnya melakukan sesuatu yang bebeda

IDE	Singkatan dari Integrated Development Environment. Merupakan teks editor dengan sejumlah alat bantu yang dapat digunakan oleh programmer dalam melakukan coding
IDLE	Singkatan dari Integrated DeveLopment Environment. Merupakan sebutan bagi IDE Phyton 3
infinite loop	Perulangan yang terus berlangsung pada saat program running. Secara umum kondisi ini tidak diharapkan dalam program
integer	Tipe data bilangan yang tidak memiliki nilai desimal melainkan harus berupa bilangan bulat
interactive mode	Mode yang digunakan saat programmer mencoba potongan program tanpa melakukan penyimpanan
local variable	Variabel yang didefinisikan dalam suatu fungsi (function) dan hanya dpat digunakan di dalam function tersebut
logical operator	Lihat 'comparative operator'
loop	Suatu kelompok code yang menjadikan program terus berulang sampai suatu kondisi terpenuhi
mathematical operator	Operator yang melaksanakan suatu fungsi matematika pada bilangan. Misalnya perkalian atau penjumlahan
method	Nama yang diberikan pada suatu fungsi dalam class
module	File Phyton yang telah tersimpan dan fungsinya dapat digunakan oleh program lain
modulus	Operator matematika yang digunakan untuk mengembalikan hasil berupa sisa dari suatu perhitungan pembagian (sisa hasil bagi). Simbolnya (%). Misal 22%7 = 1)

operator	Simbol yang membentuk fungsi sederhana pada suatu code misalnya perkalian 2 bilangan atau pembandingan diantara keduanya. Lihat juga 'comparative operator' dan 'mathematical operator'
output	Data yang dikirim dari program menuju layar display, screen ataupun printer
return	1. Nilai yang dihasilkan oleh suatu function setelah function tersebut running (termasuk keyowrd pada Phyton) 2. Disebut juga 'enter key'. Merupakan kunci 'end of line' pada keyboard
script mode	Mode penulisan code yang selanjutnya akan disimpan pada IDLE
statement	Bagian dari code yang mewakili perintah atau tindakan. Misalnya perintah untuk mencetak
string	Daya teks. Dapat disimpan pada suatu variabel
syntax error	Error yang dihasilkan pada saat komputer gagal menjalankan program karena tidak dapat mengenali format code
tkinter	Suatu paket dari class yang diimpor ke program Phyton yang selanjutnya menjadi method bagi Phyton untuk membentuk window, membentuk gambar, dan menghasilkan animasi
variable	Nama yang mengacu pada suatu tempat penyimpanan data di memori komputer
while loop	Suatu bentuk looping yang mengulang code selama suatu kondisi yang menjadi syarat bernilai 'true'

INDEKS