BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Pengumpulan Data

Berdasarkan dataset jamur yang terdapat dalam sistus *https://archive.ics.uci.edu/ml/datasets/mushroom* (UCI *Machine Learning Repository*) yang disumbangkan oleh Jeff Schlimmer pada tahun 1987 terdapat sebanyak 8124 data jamur dari keluarga *agaricus* dan *lepiota*. Data tersebut memiliki atribut sebanyak 22 dan dikelompokkan ke dalam dua kelas yakni kelas jamur dapat dimakan sebanyak 4208 dan kelas jamur beracun sebanyak 3916 jamur.

Tabel 4. 1 Hasil Data Observasi

NO	Bentuk Topi	Permukaan Topi	Warna Topi	Memar	Bau	Lampiran Lamela	Jarak Lamela	Ukuran Lamela	Warna Lamela	Bentuk Tangkai	Akar Tangkai
1	x	s	n	t	p	f	c	n	k	e	e
2	x	s	y	t	a	f	c	b	k	e	c
3	b	s	w	t	l	f	c	b	n	e	c
4	x	y	w	t	p	f	c	n	n	e	e
5	x	s	g	f	n	f	w	b	k	t	e
6	x	y	y	t	a	f	c	b	n	e	c
7	b	s	w	t	a	f	c	b	g	e	c
8	b	y	w	t	l	f	c	b	n	e	c
9	x	y	w	t	p	f	c	n	p	e	e
10	b	s	y	t	a	f	c	b	g	e	c
11	x	y	y	t	l	f	c	b	g	e	c
12	x	y	y	t	a	f	c	b	n	e	c
13	b	s	y	t	a	f	c	b	w	e	c
14	x	y	w	t	p	f	c	n	k	e	e
15	x	f	n	f	n	f	w	b	n	t	e
16	s	f	g	f	n	f	c	n	k	e	e
17	f	f	w	f	n	f	w	b	k	t	e
18	x	s	n	t	p	f	c	n	n	e	e
19	x	y	w	t	p	f	c	n	n	e	e
20	x	s	n	t	p	f	c	n	k	e	e
21	b	s	y	t	a	f	c	b	k	e	c
22	x	y	n	t	p	f	c	n	n	e	e
23	b	y	y	t	l	f	c	b	k	e	c
24	b	y	w	t	a	f	c	b	w	e	c
25	b	s	w	t	l	f	c	b	g	e	c
26	f	s	w	t	p	f	c	n	n	e	e
27	x	y	y	t	a	f	c	b	n	e	c
...	...	...	...	...	...	...	...	...	...	...	...
8124	x	s	n	f	n	a	c	b	y	e	?

Tabel 4. 1 Data Observasi (Lanjutan)

Permukaan Tangkai di atas cincin	Permukaan Tangkai di bawah cincin	Warna Tangkai di atas Cincin	Warna Tangkai di bawah Cincin	Tipe Kerudung	Warna Kerudung	Jumlah Cincin	Tipe Cincin	Warna Spora	Populasi	habitat	Kelas
s	s	w	w	p	w	o	p	k	s	u	p
s	s	w	w	p	w	o	p	n	n	g	e
s	s	w	w	p	w	o	p	n	n	m	e
s	s	w	w	p	w	o	p	k	s	u	p
s	s	w	w	p	w	o	e	n	a	g	e
s	s	w	w	p	w	o	p	k	n	g	e
s	s	w	w	p	w	o	p	k	n	m	e
s	s	w	w	p	w	o	p	n	s	m	e
s	s	w	w	p	w	o	p	k	v	g	p
s	s	w	w	p	w	o	p	k	s	m	e
s	s	w	w	p	w	o	p	n	n	g	e
s	s	w	w	p	w	o	p	k	s	m	e
s	s	w	w	p	w	o	p	n	s	g	e
s	s	w	w	p	w	o	p	n	v	u	p
s	f	w	w	p	w	o	e	k	a	g	e
s	s	w	w	p	w	o	p	n	y	u	e
s	s	w	w	p	w	o	e	n	a	g	e
s	s	w	w	p	w	o	p	k	s	g	p
s	s	w	w	p	w	o	p	n	s	u	p
...	...	...	...	...	...	...	...	...	...	...	...
s	s	o	o	p	o	o	p	o	c	l	e

Berikut keterangan inisial dari nilai atribut di atas:

1. Bentuk topi: b = bel, c = kerucut, x = cembung, f = datar, k = berujung, s = cekung
2. Permukaan topi: f = berserat , g = alur, y = bersisik, s = halus
3. Warna topi: n = coklat, b = kekuning-kuningan, c = kayu manis, g = abu-abu, r = hijau, p = merah muda, u = ungu, e = merah, w = putih, y = kuning
4. Memar: t = memar, f = tidak memar
5. Bau: a = almon, I = adas manis, c = kreosot, y = amis, f = busuk, m = apek, n = tidak memiliki bau, p = menyengat, s = pedas
6. Lampiran lamela: a = dempet, d = turun, f = bebas, n = berlekuk
7. Jarak lamela: c = dekat, w = berdesakan, d = jauh
8. Ukuran lamela: b = lebar, n = sempit
9. Warna lamela: k = hitam, n = coklat, b = kekuning-kuningan, h = coklat, g = abu-abu, r = hijau, o = oranye, p = merah muda, u = ungu, e = merah, w = putih, y = kuning
10. Bentuk tangkai: e = membesar, t = meruncing

11. Akar tangkai: b = bulat, c = tangkai, u = cangkir, e = sama, z = *rhizomorph*, r = berakar, ? = data hilang
12. Permukaan tangkai di atas cincin: f = berserat, y = bersisik, k = seperti sutra, s = halus
13. Permukaan tangkai di bawah cincin: f = berserat, y = bersisik, k = seperti sutra, s = halus
14. Warna tangkai di atas cincin: n = coklat, b = kekuning-kuningan, c = kayu manis, g = abu-abu, o = oranye, p = merah muda, e = merah, w = putih, y = kuning
15. Warna tangkai di bawah cincin: n = coklat, b = kekuning-kuningan, c = kayu manis, g = abu-abu, o = oranye, p = merah muda, e = merah, w = putih, y = kuning
16. Tipe kerudung: p = parsial, u = universal
17. Warna kerudung: n = coklat, o = oranye, w = putih, y = kuning
18. Jumlah cincin: n = tidak ada, o = satu, t = dua
19. Tipe cincin: c = jaring laba-laba, e = samar, f = terang, l = besar, n = tidak ada, p = liontin, s = berlapis, z = zona
20. Warna spora: k = hitam, n = coklat, b = kekuning-kuningan, h = coklat, r = hijau, o = oranye, u = ungu, w = putih, y = kuning
21. Populasi: a = melimpah, c = bergerombol, n = banyak, s = tersebar, v = beberapa, y = sendiri
22. Habitat: g = rerumputan, l = daun, m = padang rumput, p = jalan, u = perkotaan, w = sampah, d = kayu

Kelas: p = beracun, e = dapat dimakan

4.2 Hasil Pengolahan Data

Dari proses pengolahan data yang telah dilakukan menggunakan *Microsoft Excel*, didapatkan hasil berupa data sebanyak 8124 data dengan penghapusan satu atribut yakni atribut tipe kerudung, penghapusan atribut tersebut dikarenakan tidak adanya variasi nilai dari atribut tipe kerudung, sehingga yang awalnya terdapat 22 atribut berkurang satu menjadi 21 atribut dan nilai dari masing-masing atribut yang sudah berupa angka. Ke-21 atribut tersebut meliputi bentuk topi,

permukaan topi, warna topi, memar, bau, lampiran lamela, jarak lamela, ukuran lamela, warna lamela, bentuk tangkai, akar tangkai, permukaan tangkai di atas, permukaan tangkai di bawah, warna tangkai di atas cincin, warna tangkai di bawah cincin, warna kerudung, jumlah cincin, tipe cincin, warna spora, populasi, dan habitat. Hasil dari pengolahan data tersebut kemudian disimpan dengan nama data_jamur dalam format .csv.

Tabel 4. 2 Hasil Pengolahan Data

No	Bentuk Topi	Permukaan Topi	Warna Topi	Memar	Bau	Lampiran Lamela	Jarak Lamela	Ukuran Lamela	Warna Lamela	Bentuk Tangkai	Akar Tangkai
1	3	4	1	1	8	3	1	2	1	1	4
2	3	4	10	1	1	3	1	1	1	1	2
3	1	4	9	1	2	3	1	1	2	1	2
4	3	3	9	1	8	3	1	2	2	1	4
5	3	4	4	2	7	3	2	1	1	2	4
6	3	3	10	1	1	3	1	1	2	1	2
7	1	4	9	1	1	3	1	1	5	1	2
8	1	3	9	1	2	3	1	1	2	1	2
9	3	3	9	1	8	3	1	2	8	1	4
10	1	4	10	1	1	3	1	1	5	1	2
11	3	3	10	1	2	3	1	1	5	1	2
12	3	3	10	1	1	3	1	1	2	1	2
13	1	4	10	1	1	3	1	1	11	1	2
14	3	3	9	1	8	3	1	2	1	1	4
15	3	1	1	2	7	3	2	1	2	2	4
16	6	1	4	2	7	3	1	2	1	1	4
17	4	1	9	2	7	3	2	1	1	2	4
...	...	...	...	...	...	...	...	...	...	...	...
8124	3	4	1	2	7	1	1	2	12	1	0

Tabel 4. 2 Hasil Pengolahan Data (Lanjutan)

Permukaan Tangkai di atas cincin	Permukaan Tangkai di bawah cincin	Warna Tangkai di atas Cincin	Warna Tangkai di bawah Cincin	Warna kerudung	Jumlah Cincin	Tipe Cincin	Warna Spora	Populasi	Habitat	**Kelas**
4	4	8	8	3	2	6	1	4	5	1
4	4	8	8	3	2	6	2	3	1	2
4	4	8	8	3	2	6	2	3	3	2
4	4	8	8	3	2	6	1	4	5	1
4	4	8	8	3	2	2	2	1	1	2
4	4	8	8	3	2	6	1	3	1	2
4	4	8	8	3	2	6	1	3	3	2
4	4	8	8	3	2	6	2	4	3	2
4	4	8	8	3	2	6	1	5	1	1
4	4	8	8	3	2	6	1	4	3	2
4	4	8	8	3	2	6	2	3	1	2
4	4	8	8	3	2	6	1	4	3	2
4	4	8	8	3	2	6	2	4	1	2
4	4	8	8	3	2	6	2	5	5	1
4	1	8	8	3	2	2	1	1	1	2
4	4	8	8	3	2	6	2	6	5	2
4	4	8	8	3	2	2	2	1	1	2
...	...	...	...	...	...	...	...	...	...	...
4	4	5	5	2	2	6	6	2	2	2

Berikut keterangan nilai dari atribut di atas:

1. Bentuk topi: 1 = bel, 2 = kerucut, 3 = cembung, 4 = datar, 5 = berujung, 6 = cekung
2. Permukaan topi: 1 = berserat , 2 = alur, 3 = bersisik, 4 = halus
3. Warna topi: 1 = coklat, 2 = kekuning-kuningan, 3 = kayu manis, 4 = abu-abu, 5 = hijau, 6 = merah muda, 7 = ungu, 8 = merah, 9 = putih, 10 = kuning
4. Memar: 1 = memar, 2 = tidak memar
5. Bau: 1 = almon, 2 = adas manis, 3 = kreosot, 4 = amis, 5 = busuk, 6 = apek, 7 = tidak memiliki bau, 8 = menyengat, 9 = pedas
6. Lampiran lamela: 1 = dempet, 2 = turun, 3 = bebas, 4 = berlekuk
7. Jarak lamela: 1 = dekat, 2 = berdesakan, 3 = jauh
8. Ukuran lamela: 1 = lebar, 2 = sempit
9. Warna lamela: 1 = hitam, 2 = coklat, 3 = kekuning-kuningan, 4 = coklat, 5 = abu-abu, 6 = hijau, 7 = oranye, 8 = merah muda, 9 = ungu, 10 = merah, 11 = putih, 12 = kuning
10. Bentuk tangkai: 1 = membesar, 2 = meruncing
11. Akar tangkai: 1 = bulat, 2 = tangkai, 3 = cangkir, 4 = sama, 5 = *rhizomorph*, 6 = berakar, 0 = data hilang
12. Permukaan tangkai di atas cincin: 1 = berserat, 2 = bersisik, 3 = seperti sutra, 4 = halus
13. Permukaan tangkai di bawah cincin: 1 = berserat, 2 = bersisik, 3 = seperti sutra, 4 = halus
14. Warna tangkai di atas cincin: 1 = coklat, 2 = kekuning-kuningan, 3 = kayu manis, 4 = abu-abu, 5 = oranye, 6 = merah muda, 7 = merah, 8 = putih, 9 = kuning
15. Warna tangkai di bawah cincin: 1 = coklat, 2 = kekuning-kuningan, 3 = kayu manis, 4 = abu-abu, 5 = oranye, 6 = merah muda, 7 = merah, 8 = putih, 9 = kuning
16. Warna kerudung: 1 = coklat, 2 = oranye, 3 = putih, 4 = kuning
17. Jumlah cincin: 1 = tidak ada, 2 = satu, 3 = dua
18. Tipe cincin: 1 = jaring laba-laba, 2 = samar, 3 = terang, 4 = besar, 5 = tidak ada, 6 = liontin, 7 = berlapis, 8 = zona

19. Warna spora: 1 = hitam, 2 = coklat, 3 = kekuning-kuningan, 4 = coklat, 5 = hijau, 6 = oranye, 7 = ungu, 8 = putih, 9 = kuning
20. Populasi: 1 = melimpah, 2 = bergerombol, 3 = banyak, 4 = tersebar, 5 = beberapa, 6 = sendiri
21. Habitat: 1 = rerumputan, 2 = daun, 3 = padang rumput, 4 = jalan, 5 = perkotaan, 6 = sampah, 7 = kayu
22. Kelas: 1 = beracun, 2 = dapat dimakan

4.3 Hasil Analisis Metode

Analisis metode merupakan proses mengimplementasikan algoritma yang digunakan, dalam hal ini yakni algoritma PCA dan KNN. Dalam penelitian ini proses penerapan algoritma menggunakan bantuan bahasa pemograman python versi 3.9 dan google colab. Di mana algoritma PCA diterapkan untuk melakukan seleksi atribut sehingga atribut dari dataset yang digunakan menjadi lebih sedikit. Sedangkan algoritma KNN digunakan untuk melakukan klasifikasi.

Langkah pertama yang harus dilakukan dalam mengimplementasikan algoritma PCA dan KNN di pemograman *python* adalah mengunggah dataset yang akan digunakan. pengunggahan data bisa langsung diunggah di projek *googel colab* atau bisa juga diunggah terlebih dahulu di *google drive* yang akan di sambungkan dengan *googel colab*.

Setelah data diunggah, selanjutnya adalah mengimport *library* yang akan digunakan. *Library* yang digunakan dalam penelitian ini antara lain *numpy* untuk perhitungan algoritmanya, *pandas* untuk manajemen atau analisis datanya, serta *library matplotlib* dan *seaborn* untuk memvisualisasikan datanya. Selain itu, dalam pemograman *python* terdapat pula *library* yang memberikan kemudahan dalam proses pembuatan model *Mechine Learning* yaitu *library Scikit-learn* atau *Sklearn*. *Library* ini dapat melakukan beragam pekerjaan dalam data *science*, seperti *preprocessing* data, klasifikasi, reduksi dimensi, dan lain sebagainya. Dalam penelitian ini *library sklearn* digunakan untuk mengimport algoritma PCA, pembagian data *training* dan data *testing*, serta mengimport algoritma KNN untuk klasifikasi. Untuk melakukan import *library* di *python* dapat dilihat pada segmen program di bawah ini.

Segmen Program 4. 1 Import *Library* yang digunakan

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

from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix

pd.options.display.max_columns = 100
```

Selanjutnya adalah memanggil data yang telah diunggah sebelumnya sehingga bisa dibaca dan ditampilkan di sistem. Seperti yang telah dijelaskan di atas untuk membaca atau memanggil data di pemograman *python* menggunkan *library pandas*.

Segmen Program 4. 2 Memanggil Dataset

```
data = pd.read_csv('drive/MyDrive/Skripsi/data_jamur.csv')
data
```

	Bentuk Topi	Permukaan Topi	Warna Topi	Memar	Bau	Lampiran Lamela	Jarak Lamela	Ukuran Lamela	Warna Lamela	Bentuk Tangkai	Akar Tangkai	Permukaan Tangkai di atas cincin	Permukaan Tangkai di bawah cincin	Warna Tangkai di atas Cincin	Warna Tangkai di bawah Cincin	Warna kerudung	Jumlah Cincin	Tipe Cincin	Warna Spora	Populasi	Habitat	Kelas
0	3	4	1	1	8	3	1	2	1	1	4	4	4	8	8	3	2	6	1	4	5	1
1	3	4	10	1	1	3	1	1	1	1	2	4	4	8	8	3	2	6	2	3	1	2
2	1	4	9	1	2	3	1	1	2	1	2	4	4	8	8	3	2	6	2	3	3	2
3	3	3	9	1	8	3	1	2	2	1	4	4	4	8	8	3	2	6	1	4	5	1
4	3	4	4	2	7	3	2	1	1	2	4	4	4	8	8	3	2	2	2	1	1	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8119	5	4	1	2	7	1	1	1	12	1	0	4	4	5	5	2	2	6	3	2	2	2
8120	3	4	1	2	7	1	1	1	12	1	0	4	4	5	5	1	2	6	3	5	2	2
8121	4	4	1	2	7	1	1	1	2	1	0	4	4	5	5	2	2	6	3	2	2	2
8122	5	3	1	2	4	3	1	2	3	2	0	4	3	8	8	3	2	2	8	5	2	1
8123	3	4	1	2	7	1	1	2	12	1	0	4	4	5	5	2	2	6	6	2	2	2

8124 rows × 22 columns

Gambar 4. 1 Hasil Pemanggilan Dataset

Dari hasil pemanggilan data yang dihasilkan pada Gambar 4.1, ditampilkan data dengan jumlah total baris 8124 dan 22 kolom, di mana 21 kolom merupakan atribut dan satu kolom tersisa merupakan kelas atau labelnya.

Setelah dataset yang akan digunakan terbaca oleh sistem, selanjutnya adalah mengubah nama dari setiap kolom dengan menghapus spasi dari nama kolom yang memiliki spasi, untuk melakukan perubahan nama dapat dilihat pada segmen program 4.3. Kemudian untuk membedakan antara kolom atribut dan kolom kelas, dapat dilihat seperti pada segmen program 4.4. Di mana dalam hal ini penggunaan variabel X untuk menampung semua kolom atribut dan variabel Y untung menampung kelasnya.

Segmen Program 4. 3 Mengubah Nama Kolom

```
data.rename(columns=lambda x: x.replace(' ', ''), inplace=True)
data
```

Segmen Program 4. 4 Membedakan Antara Kolom Atribut dan Kelas

```
X=data.drop(columns='Kelas')
Y=data.Kelas
```

4.3.1 Penerapan Algoritma PCA

Setelah kolom atribut dan kelas terpisah, langkah selanjutnya adalah mengolah data atribut ke dalam algoritma PCA.

Segmen Program 4. 5 Penerapan Algoritma PCA

```
pca = PCA(n_components=21)
pca.fit(X)
x_pca = pca.fit_transform(X)
```

Sesuai dengan tahapan PCA yang telah dijelaskan pada landasan teori mengenai PCA di atas, tahap awal adalah menghitung korelasi antar dua atribut dengan persamaan varians (2.1) dan kovarians (2.2). Contoh perhitungan nilai varians dan kovarians pada dataset dalam penelitian ini akan dicontohkan pada atribut ke-1 dan 2, yaitu bentuk topi (x) dan permukaan topi (y).

$$Var(x) = \frac{\sum(3-3.491)^2+(3-3.491)^2\ldots(5-3.491)^2+(3-3.491)^2}{(8124-1)} = 0.812$$

$$Cov(x,y) = \frac{\sum(3-3.491)(4-2.742)+(3-3.491)(4-2.742)\ldots(3-3.491)(4-2.742)}{(8124-1)} = -0.007$$

Dari perhitungan nilai varians dan kovarians maka akan didapatkan sebuah matrik berukuran $n \times n$ yang dinamakan *covariance matrix*. Matrik kovarians terdiri dari perhitungan nilai varians kovarians, yang mana dalam penelitian ini matrik kovarians yang terbentuk berukuran 21×21 yaitu sebanyak atribut yang ada pada dataset. Untuk menghitung matrik kovarians di pemograman *python* dapat dilihat pada segmen program di bawah ini.

Segmen Program 4. 6 Menghitung *Covariance Matrix* di *Python*

```
covmat = pca.get_covariance()
covmat = pd.DataFrame(covmat).round(4)
covmat
```

Tabel 4. 3 Hasil Perhitungan *Covariance Matrix*

Atribut	1	2	3	4	5	6	7	8	…	21
1	0.8123	-0.0075	-0.5482	0.0887	0.4489	0.0092	-0.0204	0.1078	…	0.3061
2	-0.0075	1.3915	-0.0931	0.0114	-0.2475	-0.0606	-0.0417	0.1501	…	-0.572
3	-0.5482	-0.0931	11.8638	-0.0586	-2.1789	0.2103	0.0295	-0.1469	…	-0.7821
4	0.0887	0.0114	-0.0586	0.2429	0.0775	-0.0215	0.0543	0.0842	…	-0.3809
5	0.4489	-0.2475	-2.1789	0.0775	3.935	-0.0582	0.0834	0.0341	…	0.6328
6	0.0092	-0.0606	0.2103	-0.0215	-0.0582	0.1007	0.0084	0.0157	…	0.0927
7	-0.0204	-0.0417	0.0295	0.0543	0.0834	0.0084	0.1354	-0.0184	…	-0.3706
8	0.1078	0.1501	-0.1469	0.0842	0.0341	0.0157	-0.0184	0.2137	…	0.1027
9	-0.2084	-0.4847	-0.2338	-0.5743	0.4967	-0.0856	-0.0464	-0.5099	…	1.5649
10	0.111	0.0217	-0.41	-0.0243	0.3387	0.0293	0.0148	0.0491	…	0.3
11	-0.1544	-0.1783	0.2163	-0.1295	-0.3364	0.0581	0.2015	-0.183	…	-1.2613
12	-0.0521	0.0148	-0.0479	-0.1553	-0.0718	-0.0215	-0.0893	0.0341	…	0.5979
13	-0.0539	-0.0001	-0.0808	-0.1386	0.1038	-0.0209	-0.0775	0.0351	…	0.6068
…	…	…	…	…	…	…	…	…	…	…
21	0.3061	-0.572	-0.7821	-0.3809	0.6328	0.0927	-0.3706	0.1027	…	6.4044

Dari hasil perhitungan matrik kovarians maka dapat dicari *eigenvalue* dan *eigenvector* dari matrik tersebut. Pada segmen program 4.7 diperlihatkan kode pada pemograman *python* untuk mencari *eigenvalue* dan *eigenvector* dari matrik kovarians. Begitu juga cara menampilkan hasil *eigenvalue* yang telah didapat.

Segmen Program 4. 7 Perhitungan *Eigenvalue* dan *Eigenvector* di *Python*

```
eigval, eigvec = np.linalg.eig(covmat)
print('==========================Nilai Eigen==========================')
print(eigval)
```

```
==============================Nilai Eigen==============================
[1.39860453e+01 1.30107756e+01 9.71922640e+00 7.90888452e+00
 5.75990115e+00 3.51001444e+00 2.01025908e+00 1.70156505e+00
 1.10510011e+00 9.76580519e-01 6.67030698e-01 6.21982026e-01
 5.20643129e-01 3.92158515e-01 1.62516832e-01 1.19252579e-01
 6.25666450e-03 1.34202893e-02 2.68160016e-02 8.52465688e-02
 9.41245092e-02]
```

Gambar 4. 2 Hasil Perhitungan *Eigenvalue* di *Python*

Berdasarkan hasil perhitungan *eigenvalue* inilah suatu *principal component* terbentuk. Pada Gambar 4.2 di atas terdapat hasil perhitungan *eigenvalue* yang berjumlah 21, yang mana dapat diartikan terdapat 21 *principal component* yang terbentuk. Nilai *eigen* dibentuk berdasarkan diagonal matrik kovarian menggunakan persamaan 2.3. Pada tahap ini masih belum terjadi reduksi dimensi karena atribut awal dan principal component totalnya masih sama-sama 21. Untuk melakukan reduksi dimensi tahap selanjutnya yang harus dilakukan adalah menghitung nilai proporsi varians dari masing-masing *eigenvalue* atau *principal component*. Perhitungan proporsi varians ini dilakukan untuk mengetahui persentase nilai dari masing-masing komponen yang telah terbentuk berdasarkan *eigenvalue*. Sehingga dengan mengetahui nilai persentase komponen tersebut dapat diketahui berapa komponen yang akan digunkan berdasarkan nilai proporsi varians yang diinginkan. Nilai varians kovarians didapat dari hasil perhitungan penjumlahan nilai diagonal matriks dari masing-masing atribut dataset jamur seperti berikut:

$$\begin{aligned} VarianceCovarian = {} & 0.8123 + 1.3915 + 11.8638 + 0.2429 + 3.935 + \\ & 0.1007 + 0.1354 + 0.2137 + 11.1717 + 0.2455 + \\ & 2.1186 + 0.6637 + 0.7575 + 4.5963 + 4.8163 + \\ & 0.0589 + 0.0735 + 3.246 + 7.9824 + 1.5677 + \\ & 6.4044 = 62.39 \end{aligned}$$

Setelah hasil dari perhitungan dari varians kovarians didapat kemudian dilanjutkan dengan perhitungan nilai proporsi varians dari masing-masing *principal component* menggunakan persamaan 2.4 seperti berikut:

1. $PC1(\%) = \frac{NilaiEigen\ PC1}{VarianceCovarian} \times 100\% = \frac{13.98}{62.39} \times 100\% = 22.41\%$
2. $PC2(\%) = \frac{NilaiEigen\ PC2}{VarianceCovarian} \times 100\% = \frac{13.01}{62.39} \times 100\% = 20.85\%$
3. $PC3(\%) = \frac{NilaiEigen\ PC3}{VarianceCovarian} \times 100\% = \frac{9.71}{62.39} \times 100\% = 15.57\%$
4. $PC4(\%) = \frac{NilaiEigen\ PC4}{VarianceCovarian} \times 100\% = \frac{7.90}{62.39} \times 100\% = 12.67\%$
5. $PC5(\%) = \frac{NilaiEigen\ PC5}{VarianceCovarian} \times 100\% = \frac{5.75}{62.39} \times 100\% = 9.23\%$

Segmen Program 4.8 di bawah ini merupakan kode di pemograman *python* untuk menghitung total varians kovarians, menghitung proporsi varians yang diberi nama variabel var_exp atau dalam pemograman *python* disebut dengan *explained variance*, serta perhitungan *cumulative explained variance* atau akumulasi dari nilai *explained variance.* Sehingga diperoleh hasil seperti pada Gambar 4.3.

Segmen Program 4. 8 Perhitungan Proporsi Varians di *Python*

```
var_cov = np.sum(eigval)
print('==================Varians Kovarians==================')
print(var_cov)
var_exp = (eigval/var_cov)*100
print('==================Proporsi Varians===================')
print(var_exp)
cum_var_exp=np.cumsum(var_exp)
print('==================Cumulative Varians=================')
print(cum_var_exp)
print()
```

```
==============================Varians Kovarians==============================
62.397800000000046
==============================Proporsi Varians===============================
[2.24143244e+01 2.08513371e+01 1.55762325e+01 1.26749413e+01
 9.23093627e+00 5.62522146e+00 3.22168263e+00 2.72696322e+00
 1.77105621e+00 1.56508806e+00 1.06899714e+00 9.96801211e-01
 8.34393406e-01 6.28481316e-01 2.60452824e-01 1.91116641e-01
 1.00270595e-02 2.15076322e-02 4.29758768e-02 1.36617908e-01
 1.50845878e-01]
==============================Cumulative Varians=============================
[ 22.41432437  43.26566147  58.84189397  71.51683526  80.74777153
  86.37299299  89.59467562  92.32163884  94.09269504  95.65778311
  96.72678025  97.72358146  98.55797486  99.18645618  99.44690901
  99.63802565  99.64805271  99.66956034  99.71253621  99.84915412
 100.        ]
```

Gambar 4. 3 Hasil Perhitungan Proporsi Varians

Untuk tampilan yang lebih rapi, hasil dari perhitungan nilai *eigen*, proporsi varians, dan kumulatif varians disajikan dalam bentuk tabel seperti yang dapat dilihat pada Tabel 4.4.

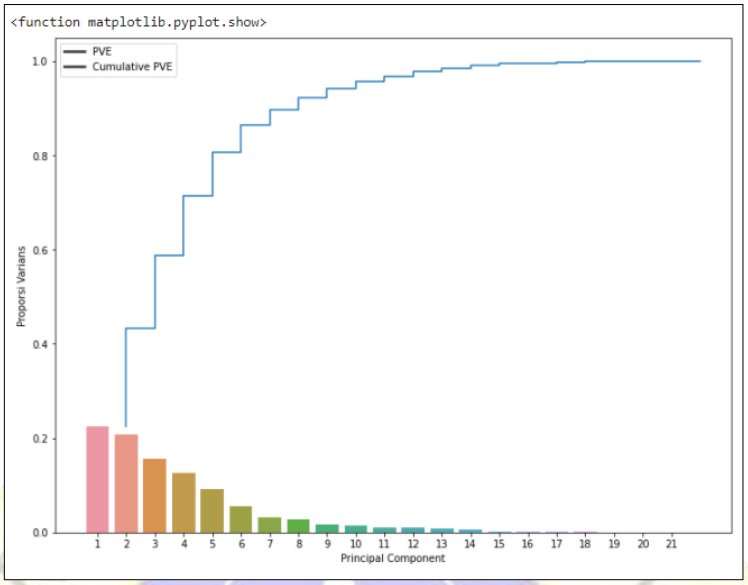
Tabel 4. 4 Hasil Persentase Varians

Principal Component	Nilai Eigen	Proporsi Varians	Cumulative Varians
1	13.986071	22.414324	22.414324
2	13.010829	20.851337	43.265661
3	9.71927	15.576232	58.841894
4	7.908812	12.674941	71.516835
5	5.759899	9.230936	80.747772
6	3.509955	5.625221	86.372993
7	2.010305	3.221683	89.594676
8	1.701546	2.726963	92.321639
9	1.10512	1.771056	94.092695
10	0.976558	1.565088	95.657783
11	0.667008	1.068997	96.72678
12	0.621986	0.996801	97.723581
13	0.520625	0.834393	98.557975
14	0.392163	0.628481	99.186456
15	0.16253	0.260453	99.446909
16	0.119248	0.191117	99.638026
17	0.094102	0.010027	99.648053
18	0.085302	0.021508	99.66956
19	0.026783	0.042976	99.712536
20	0.013449	0.136618	99.849154
21	0.006215	0.150846	100

Bentuk dari hasil perhitungan nilai *eigen*, proporsi varians, serta hasil dari kumulatif proporsi varians dapat divisualisasikan dalam pemograman *python* menggunakan kode seperti pada segmen program di bawah ini. Sehingga dihasilkan visualisasi data seperti pada Gambar 4.14. hasil visualisasi data tersebut digambarkan dalam bentuk diagram batang untuk nilai proporsi varians dan garis step untuk kumulatif proporsi varians.

Segmen Program 4. 9 Prosedur Visualisasi PCA di *Python*

```
plt.figure(figsize=(12,9))
sns.barplot(x='Principal Component', y='Proporsi Varians', data=PVE)
plt.step(x='Principal Component', y='Cumulative Varians', data=PVE)
plt.legend(['PVE','Cumulative PVE'])
plt.show
```



Gambar 4. 4 Hasil Visualisasi PCA

Tahap selanjutnya adalah memilih beberapa *principal component* berdasarkan hasil kumulatif proporsi varians. Pada penelitian ini kumulatif proporsi varians yang diambil adalah sebesar 89.59%. persentase tersebut diperoleh dari penjumlahan nilai proporsi varians dari *principal component* ke-1 sampai dengan *principal component* ke-7 sehingga diperoleh sejumlah *principal component* sebanyak 7 seperti yang dapat dilihat pada Tabel 4.5.

Tabel 4. 5 Persentase Varians Berdasarkan *Principal Component* Terpilih

Principal Component	Nilai Eigen	Proporsi Varians	Cumulative Varians
1	13.986071	22.414324	22.414324
2	13.010829	20.851337	43.265661
3	9.71927	15.576232	58.841894
4	7.908812	12.674941	71.516835
5	5.759899	9.230936	80.747772
6	3.509955	5.625221	86.372993
7	2.010305	3.221683	89.594676
***Variance Threshold* = 89.59%**			

Berdasarkan Tabel 4.5, persentase proporsi varians PC ke-1 hanya dapat menyerap informasi sebesar 22.41% dari total varians kovarians data asli, kemudian jika ditambah dengan proporsi varians PC ke-2 hanya dapat menyerap informasi sebesar 43.26% dari total varians kovarians data asli. Sehingga pemilihan *principal component* dilanjutkan sampai PC ke-7 dengan

total persentase varians sebesar 89.59%. Persentase ini dinilai cukup tinggi dalam menyerap informasi dari total varians kovarians data asli. Hasil pemilihan 7 *principal component* inilah yang disebut terjadinya reduksi dimensi. Di mana data yang awalnya berdimensi 21 kini berubah menjadi 7 dimensi saja dengan perhitungan proporsi varians kumulatif yang menghasilkan persentase varians sebesar 89.59%.

Setelah hasil reduksi dimensi diperoleh, tahap selanjutnya adalah menghitung bobot atribut berdasarkan *eigenvalue* dengan menggunakan persamaan *eigenvector* 2.5. Perhitungan bobot atribut ini dilakukan untuk mengetahui atribut apa saja dari data asli yang termasuk ke dalam sejumlah *principal component* yang telah dipilih dengan total persentase varians sebesar 89.59%.

Untuk menghitung *eigenvector* pada pemograman *python* dapat dilihat pada segmen program 4.7. Di mana *eigenvalue* dihitung berdasarkan kovarians matriks *(cov_mat)* dan disimpan dengan variabel *eigval*, kemudian hasil perhitungan bobot atribut berdasarkan *eigenvector* disimpan ke dalam variabel *eigvec*. Tabel 4.6 adalah hasil perhitungan bobot atribut menggunakan *eigenvector*.

Tabel 4. 6 Hasil Perhitungan *Eigenvector*

Atribut	PC1	PC2	PC3	PC4	PC5	PC6	PC7	...	PC21
1	-0.046	0.065	-0.057	0.001	-0.026	0.103	-0.130	...	-0.000
2	-0.103	-0.007	0.021	-0.142	-0.047	-0.139	0.101	...	0.095
3	0.156	-0.880	-0.311	-0.145	-0.151	0.241	0.016	...	-0.008
4	-0.075	0.003	-0.056	-0.011	0.071	0.032	-0.082	...	0.369
5	-0.024	0.262	-0.020	-0.013	-0.031	0.893	0.319	...	-0.025
6	-0.001	-0.017	0.002	0.005	-0.028	0.013	-0.065	...	0.317
7	-0.014	-0.013	0.046	-0.030	0.032	0.056	-0.054	...	0.730
8	-0.066	0.015	-0.026	-0.031	-0.072	-0.014	-0.029	...	0.185
9	0.732	0.228	-0.081	-0.590	0.152	0.004	-0.139	...	0.007
10	-0.030	0.036	0.018	0.013	-0.078	0.079	-0.050	...	0.292
11	0.052	-0.109	0.330	0.083	0.166	0.105	-0.327	...	0.004
12	0.046	0.014	0.011	0.007	-0.129	-0.078	0.181	...	0.108
13	0.035	0.022	-0.008	0.018	-0.131	-0.036	0.262	...	0.028
...	...	...	...	...	...	...	...	...	...
21	0.260	0.217	-0.28	0.30	-0.734	0.005	-0.323	...	0.005

Pada penelitian ini hasil seleksi atribut diperoleh dari pemilihan atribut yang memiliki bobot nilai tertinggi dari hasil perhitungan *eigenvector*. Hasil seleksi atribut dapat dilihat pada tabel 4.7.

Tabel 4. 7 Hasil Seleksi Atribut Dataset Jamur Menggunakan PCA

No.	Principal Component	Bobot Nilai	Atribut
1	PC1	0.732	Warna Lamela
2	PC2	0.880	Warna Topi
3	PC3	0.529	Warna Spora
4	PC4	0.590	Warna Lamela
5	PC5	0.734	Habitat
6	PC6	0.893	Bau
7	PC7	0.659	Tipe Cincin

Berdasarkan tabel di atas diperoleh hasil seleksi atribut sebanyak 6 atribut dari PC 1 sampai PC 7 yang mana *principal component* tersebut memiliki korelasi cukup besar terhadap pembentukaan atribut asli dengan total persentase varians kovarians sebesar 89.59% dari dataset jamur. Pada PC 1 dan PC 4 bobot nilai tertinggi sama-sama dimiliki oleh atribut warna lamela dengan bobot nilai sebesar 0.732 dan 0.590, sehingga dari 7 *principal component* hanya ada 6 atribut dari hasil seleksi atribut karena terdapat 2 *principal component* yang atributnya sama. Ke-6 atribut terpilih inilah yang akan diklasifikasikan menggunakan algoritma KNN yang diharapkan dengan atribut yang lebih sedikit ini dapat mengoptimalkan kinerja algoritma KNN.

4.3.2 Penerapan Algoritma KNN

Berdasarkan hasil dari seleksi atribut yang telah didapat menggunakan algoritma PCA di atas. Atribut-atribut yang telah terpilih akan diklasifikasikan menggunakan algoritma KNN. Untuk melakukan klasifikasi di pemograman

python terlebih dahulu memanggil atribut yang telah diseleksi menggunakan PCA dengan kode seperti pada segmen program di bawah ini.

Segmen Program 4. 10 Pemanggilan Data Hasil Seleksi Atribut PCA

```
atribut_pca = ['WarnaLamela', 'WarnaTopi','WarnaSpora',
               'Habitat', 'Bau', 'TipeCincin']
atribut_pca = data.loc[:, atribut_pca].values
atribut_pca = pd.DataFrame(atribut_pca)
atribut_pca
```

	0	1	2	3	4	5
0	1	1	1	5	8	6
1	1	10	2	1	1	6
2	2	9	2	3	2	6
3	2	9	1	5	8	6
4	1	4	2	1	7	2
...	...	...	...	...	...	...
8119	12	1	3	2	7	6
8120	12	1	3	2	7	6
8121	2	1	3	2	7	6
8122	3	1	8	2	4	2
8123	12	1	6	2	7	6

8124 rows × 6 columns

Gambar 4. 5 Hasil Pemanggilan Atribut PCA

Hasil seleksi atribut menggunakan algoritma PCA yakni meliputi warna lamela, warna topi, warna spora, habitat, bau, dan tipe cincin, ditampung dalam variabel atribut_pca. Yang mana atribut_pca ini diambil dari data awal yang diberi nama variabel data. Sehingga dihasilkan data baru dengan 6 atribut sebanyak 8124 data.

Setelah data baru berhasil terbaca oleh sistem, selanjutnya adalah melakukan pembagian data. Di mana data baru tersebut dibagi ke dalam dua jenis, yang pertama data *training* sebagai data pembelajaran untuk mesin dan yang kedua data *testing* untuk dijadikan sebagai data uji untuk melihat akurasi yang dihasilkan. Berikut kode untuk membagi data di pemograman *python*.

Segmen Program 4. 11 Pembagian Data *Training* dan Data *Testing*

```
X_train_PCA, X_test_PCA, y_train, y_test = train_test_split(
    atribut_pca, Y, test_size = 0.25, random_state=42, stratify=Y)
X_train_PCA.shape, X_test_PCA.shape, y_train.shape, y_test.shape
```

Berdasarkan Segmen Program 4.11 di atas, data *testing* diambil sebanyak 25% dari dataset, yang artinya 75% dari dataset dijadikan sebagai data *training*. Sehingga dihasilkan data *training* sebanyak 6093 dan *testing* sebanyak 2031. Berikut hasil dari pembagian data ke dalam data *training* dan data *testing*.

Tabel 4. 8 Data *Training*

No.	No. Data	Warna Lamela	Warna Topi	Warna Spora	Bau	Habitat	Tipe Cincin	Kelas
1	977	2	1	1	7	7	6	2
2	2029	1	9	1	3	2	6	2
3	1600	2	4	1	7	7	6	2
4	6583	3	1	8	4	9	2	1
5	1579	2	1	1	1	8	6	1
6	7174	3	1	8	2	5	2	1
7	2419	8	1	1	7	7	6	2
8	3121	11	4	2	7	7	6	2
9	666	11	1	2	1	1	6	2
10	3943	8	8	2	7	7	6	2
11	2932	8	1	1	7	7	6	2
12	7811	12	1	9	2	7	6	2
13	883	2	9	1	1	2	6	2
...	...	...	...	...	...	...	...	...
6093	1973	8	1	1	7	7	6	2

Tabel 4. 9 Data *Testing*

No.	No. Data	Warna Lamela	Warna Topi	Warna Spora	Bau	Habitat	Tipe Cincin	Kelas
1	4715	4	10	4	4	5	4	1
2	3115	8	1	1	7	7	6	2
3	5123	5	4	4	7	5	4	1
4	3316	11	4	2	7	7	6	2
5	2427	9	8	2	7	7	6	2
6	5329	11	9	4	1	5	6	1
7	369	2	4	1	1	7	2	2
8	7609	8	4	8	1	7	6	2
9	2365	8	1	1	7	7	6	2
10	3114	11	4	2	7	7	6	2
11	5044	11	2	4	1	5	6	1
12	4393	5	4	4	7	5	4	1
13	5600	3	1	8	2	4	2	1
...	...	...	...	...	...	...	...	...
2031	6043	3	8	8	4	4	2	1

Selanjutnya adalah dilakukannya proses klasifikasi menggunakan algoritma KNN. Tahap pertama proses klasifikasi KNN adalah menentukan nilai K terlebih dahulu. Dalam hal ini di contohkan nilai k=7. Selanjutnya adalah menghitung jarak antara data *testing* dan data *training* menggunakan persamaan 2.6 dan 2.7.

Berikut contoh perhitungan untuk menghitung jarak pada tabel data *testing* 4.9 nomor 1. Dalam hal ini contoh perhitungan jarak yang diperlihatkan menggunakan metode perhitungan jarak *euclidean distance*.

Tabel 4. 10 Perhitungan Jarak antara Data *Training* dan Data *Testing*

No.	Perhitungan Jarak (*Euclidean Distence*)	Hasil
1	$\sqrt{(2-4)^2+(1-10)^2+(1-4)^2+(7-4)^2+(7-5)^2+(6-4)^2}$	10.53565
2	$\sqrt{(1-4)^2+(9-10)^2+(1-4)^2+(3-4)^2+(2-5)^2+(6-4)^2}$	5.744563
3	$\sqrt{(2-4)^2+(4-10)^2+(1-4)^2+(7-4)^2+(7-5)^2+(6-4)^2}$	8.124038
4	$\sqrt{(3-4)^2+(1-10)^2+(8-4)^2+(4-4)^2+(9-5)^2+(2-4)^2}$	10.86278
5	$\sqrt{(2-4)^2+(1-10)^2+(1-4)^2+(1-4)^2+(8-5)^2+(6-4)^2}$	10.77033
…	…	…
6093	$\sqrt{(8-4)^2+(1-10)^2+(1-4)^2+(7-4)^2+(7-5)^2+(6-4)^2}$	11.09054

Langkah selanjutnya adalah mengurutkan hasil perhitungan jarak dari yang terkecil hingga yang terbesar, di mana jarak yang dengan nilai terkecil tersebut disebut sebagai tetangga terdekat dari data *testing*.

Tabel 4. 11 Hasil Pengurutan Jarak Terkecil Hingga Terbesar

No.	No. Data	Warna Lamela	Warna Topi	Warna Spora	Bau	Habitat	Tipe Cincin	Kelas	Jarak	Urutan Jarak
60	4789	4	10	4	4	5	4	1	0	1
114	5030	4	10	4	4	5	4	1	0	2
334	5047	4	10	4	4	5	4	1	0	3
369	4097	4	10	4	4	5	4	1	0	4
422	5046	4	10	4	4	5	4	1	0	5
466	3001	4	10	4	4	5	4	1	0	6
517	4816	4	10	4	4	5	4	1	0	7
574	4716	4	10	4	4	5	4	1	0	8
855	4773	4	10	4	4	5	4	1	0	9
1163	4874	4	10	4	4	5	4	1	0	10
1297	4562	4	10	4	4	5	4	1	0	11
1321	3967	4	10	4	4	5	4	1	0	12
1347	3956	4	10	4	4	5	4	1	0	13
…	…	…	…	…	…	…	…	…	…	…
5990	7537	12	1	9	2	7	6	2	13.49074	6093

Berdasarkan hasil pengurutan jarak dari yang terkecil hingga yang terbesar pada Tabel 4.11 dapat dilihat pada pengurutan jarak dari jarak ke-1 sampai ke-

7 didominasi oleh kelas 1. Sehingga dapat disimpulkan bahwa data *testing* pada nomor 1 yang baru saja diujikan masuk ke dalam kelas 1 atau jamur beracun.

Untuk melakukan klasifikasi menggunakan algoritma KNN di pemograman *python* dapat dilihat pada Segmen Program di bawah ini.

Segmen Program 4. 12 Penerapan Algoritma KNN di *Python*

```
knn_pca = KNeighborsClassifier(n_neighbors = 3,
                               metric = 'euclidean')
knn_pca.fit(X_train_PCA,y_train)
y_pred_pca = knn_pca.predict(X_test_PCA)
akurasi_knn_pca = round(knn_pca.score(X_test_PCA,y_test),4)
print("Test score after PCA",akurasi_knn_pca)
```

Pada Segmen Program 4.12 di atas algoritma KNN disimpan di dalam variabel dengan nama knn_pca. Di mana pada gambar tersebut dapat dilihat penerapan KNN dengan nilai n_neighbors (K=3) dan perhitungan jarak menggunakan metode *euclidean distance.* Sedangkan untuk menggunakan perhitungan jarak menggunakan metode *manhattan distance* dapat mengganti kode *metric euclidean* pada gambar menjadi *metric manhattan*. Kemudian hasil dari data *training* di atas dijadikan sebagai pembelajaran terhadap algoritma KNN. Setelah dilakukan pembelajaran terhadap algoritma KNN, selanjutnya adalah melakukan prediksi terhadap data *testing* menggunakan algoritma KNN. selanjutnya adalah melihat sekor akurasi yang diperoleh dari hasil klasifikasi menggunkan algoritma KNN.

Untuk mengetahui berapa nilai yang salah dan yang benar dari hasil prediksi yang telah dilakukan dapat menggunkan model *confusion matrix.* Dalam pemograman *python* penulisan kode untuk menerapkan model *confusion matrix* dapat dilihat pada segmen program di bawah ini.

Segmen Program 4. 13 Penerapan Model *Confusion Matrix* di *Python*

```
cm_pca = confusion_matrix(y_test, y_pred_pca)
print('===Hasil Confusion Matriks===')
cm_pca = pd.DataFrame(cm_pca)
print(cm_pca)
```

Setelah melakukan import *library confusion matrix* pada Segmen Program 4.1, langkah selanjutnya adalah melakukan pemanggilan model *confusion*

matrix untuk melakukan pengujian berdasarkan data *testing* kelas dan hasil prediksi kelas yang diperoleh dari hasil klasifikasi KNN. Sehingga dihasilkan nilai dalam bentuk tabel seperti pada Tabel 2.1.

Selanjutnya peneliti juga menerapkan algoritma KNN pada data awal yang digunakan dalam penelitian ini sebelum dilakukannya seleksi atribut. Hal ini dilakukan untuk membandingkan hasil akurasi yang diperoleh dari klasifikasi jamur dengan metode KNN konvensiaonal dengan metode KNN+PCA. Proses yang dilakukan dalam klasifikasi jamur menggunakan KNN konvensional sama dengan proses ketika menerapkan algoritma KNN pada data jamur yang atributnya telah diseleksi menggunakan algoritma PCA di atas. Hanya saja pada pembagian data *training* dan data *testing* data atribut yang dibagi adalah data awal yang atributnya masih berjumlah 21 atribut bukan data yang atributnya telah diseleksi menggunakan algoritma PCA sehingga menghasilkan 6 atribut terpilih yang korelasinya paling besar dalam membentuk 7 *principal component.*

4.4 Hasil Pengujian

Pengujian dilakukan berdasarkan penerapan algoritma KNN konvensional dan KNN+PCA dalam mengklasifikasikan data jamur dengan membandingkan dua metode perhitungan jarak yakni *euclidean* dan *manhattan distance*. Tujuan dari pengujian ini adalah untuk mengetahui hasil akurasi yang diperoleh dari penerapan algoritma KNN+PCA dan KNN konvensional dalam mengklasifikasikan data jamur. Pengujian dilakukan K yang berbeda-beda sebanyak 5 kali dari masing-masing metode perhitungan jarak yang di terapkan ke dalam algoritma KNN konvensional dan KNN+PCA. K yang digunakan dalam penelitian ini adalah K dengan nilai ganjil, hal ini dilakukan untuk memudahkan penarikan kesimpulan dengan tidak adanya hasil yang imbang dari banyaknya kelas yang masuk ke dalam tetangga terdekat. Adapun percobaan nilai K yang digunakan adalah K=3, K=5, K=7, K=9, dan K=11.

Berdasarkan hasil 5 kali pengujian dengan nilai K yang berbeda-beda meliputi K=3, K=5, K=7, K=9, K=11 didapat hasil akurasi pengujian KNN+PCA seperti pada tabel di bawah ini.

Tabel 4. 12 Hasil Pengujian KNN+PCA

No.	Nilai K	*Euclidean Distance*					*Manhattan Distance*				
		TP	TN	FP	FN	Akurasi	TP	TN	FP	FN	Akurasi
1	K = 3	979	1052	0	0	100%	979	1052	0	0	100%
2	K = 5	979	1050	2	0	99.90%	979	1050	2	0	99.90%
3	K = 7	978	1050	2	1	99.85%	979	1050	2	0	99.90%
4	K = 9	976	1047	5	3	99.61%	979	1050	2	0	99.90%
5	K = 11	976	1047	5	3	99.61%	979	1050	2	0	99.90%

Selanjutnya pengujian dilakukan kembali untuk mengetahui hasil akurasi penerapan metode KNN konvensional dengan nilai K yang sama seperti pada pengujian KNN+PCA. Berikut hasil akurasi yang diperoleh dari penerapan KNN konvensional.

Tabel 4. 13 Hasil Pengujian KNN Konvensional

No.	Nilai K	*Euclidean Distance*					*Manhattan Distance*				
		TP	TN	FP	FN	Akurasi	TP	TN	FP	FN	Akurasi
1	K = 3	979	1052	0	0	100%	979	1052	0	0	100%
2	K = 5	979	1052	0	0	100%	979	1052	0	0	100%
3	K = 7	979	1052	0	0	100%	979	1052	0	0	100%
4	K = 9	979	1052	0	0	100%	979	1052	0	0	100%
5	K = 11	979	1050	2	0	99.90%	979	1052	0	0	100%

Keterangan tabel pengujian:

TP : Jumlah data dengan kelas aktual jamur beracun dan kelas prediksinya jamur beracun.

TN : Jumlah data dengan kelas aktual jamur pangan dan kelas prediksinya jamur pangan.

FP : Jumlah data dengan kelas aktual jamur beracun dan kelas prediksinya jamur pangan.

FN : Jumlah data dengan kelas aktual jamur pangan dan kelas prediksinya jamur beracun.

Pada tabel 4.12 hasil pengujian klasifikasi data jamur menggunakan algoritma KNN dan PCA dengan membandingkan dua metode perhitungan jarak yakni *euclidean* dan *manhattan disatance* menunjukkan nilai akurasi yang sangat baik, di mana akurasi yang paling tinggi didapat sebesar 100% pada K=3 dengan nilai TP = 979, nilai TN = 1052, nilai FP = 0, dan nilai FN = 0, kemudian pada pengujian K=5 akurasi yang didapat sebesar 99.90%. Selanjutnya pada K=7, 9,

dan 11 dengan menerapkan kedua metode perhitungan jarak yang digunakan didapatkan hasil akurasi yang berbeda, di mana penerapan metode *manhattan* menghasilkan nilai akurasi yang lebih tinggi dari pada penerapan metode *euclidean*. Selisih akurasi yang di dapat pada penerapan kedua metode perhitungan jarak tersebut adalah 0.29% dari nilai akurasi paling rendah antara metode *manhattan* sebesar 99.90% dan akurasi *euclidean* sebesar 99.61%.

Selanjutnya pada tabel pengujian 4.13 di atas dapat dilihat bahwa hasil akurasi yang diperoleh menggunakan algoritma KNN konvensional memiliki nilai akurasi yang sangat baik pula yakni sebesar 100% dari pengujian dengan nilai K yang telah ditentukan menggunakan metode perhitungan jarak manhattan distance. Pada penerapan metode euclidean distance hasil yang diperoleh juga memiliki akurasi yang baik, namun ada sedikit perbedaan dari penerapan kedua metode perhitungan jarak yang digunakan, di mana pada K=11 akurasi yang diperoleh metode euclidean lebih rendah dari pada hasil akurasi menggunakan metode manhattan yakni sebesar 99.90%.