

# Modul Praktikum Kecerdasan Buatan



Rolly Maulana Awangga  
0410118609

Applied Bachelor of Informatics Engineering  
Program Studi D4 Teknik Informatika

Applied Bachelor Program of Informatics Engineering  
*Politeknik Pos Indonesia*

Bandung 2019

‘Jika Kamu tidak dapat menahan lelahnya belajar,  
Maka kamu harus sanggup menahan perihnya Kebodohan.’  
Imam Syafi’i

## **Acknowledgements**

Pertama-tama kami panjatkan puji dan syukur kepada Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga Buku Pedoman Tingkat Akhir ini dapat diselesaikan.

## **Abstract**

Buku Pedoman ini dibuat dengan tujuan memberikan acuan, bagi mahasiswa Tingkat Akhir dan dosen Pembimbing. Pada intinya buku ini menjelaskan secara lengkap tentang Standar pengerjaan Intership dan Tugas Akhir di Program Studi D4 Teknik Informatika, dan juga mengatur mekanisme, teknik penulisan, serta penilaiannya. Dengan demikian diharapkan semua pihak yang terlibat dalam aktivitas Bimbingan Mahasiswa Tingkat Akhir berjalan lancar dan sesuai dengan standar.

# Contents

<b>1</b>	<b>Mengenal Kecerdasan Buatan dan Scikit-Learn</b>	<b>1</b>
1.1	Teori . . . . .	1
1.2	Instalasi . . . . .	2
1.3	Penanganan Error . . . . .	2
1.4	Ahmad Syafrizal Huda/1164062 . . . . .	2
1.4.1	Teori . . . . .	2
1.4.2	Instalasi . . . . .	4
1.4.2.1	Instalasi Library Scikit dari Anaconda . . . . .	4
1.4.2.2	Mencoba Loading an example Dataset . . . . .	4
1.4.2.3	Learning and Predicting . . . . .	5
1.4.2.4	Model Presistence . . . . .	5
1.4.2.5	Conventions . . . . .	7
1.4.3	Penanganan eror . . . . .	10
1.4.3.1	ScreenShoot Error . . . . .	10
1.4.3.2	Tuliskan Kode Error dan Jenis Erornya . . . . .	11
1.4.3.3	Solusi Pemecahan Masalah Error . . . . .	11
<b>2</b>	<b>Related Works</b>	<b>20</b>
2.1	Ahmad Syafrizal Huda/1164062 . . . . .	20
2.1.1	Teori . . . . .	20
2.1.2	Scikit-learn . . . . .	22
2.1.3	Penanganan Error . . . . .	28
<b>3</b>	<b>Methods</b>	<b>35</b>
3.1	Ahmad Syafrizal Huda / 1164062 . . . . .	35
3.1.1	Teori . . . . .	35
3.1.2	Praktek Program . . . . .	38
3.1.3	Penanganan Error . . . . .	41

<b>4</b>	<b>Experiment and Result</b>	<b>59</b>
4.1	Experiment . . . . .	59
4.2	Result . . . . .	59
4.3	Ahmad Syafrizal Huda/1164062 . . . . .	59
4.3.1	Teori . . . . .	59
4.3.2	Praktek Program . . . . .	62
4.3.3	Penanganan Error . . . . .	65
<b>5</b>	<b>Conclusion</b>	<b>69</b>
5.1	Ahmad Syafrizal Huda/1164062 . . . . .	69
5.1.1	Teori . . . . .	69
5.1.2	Praktek Program . . . . .	72
<b>6</b>	<b>Discussion</b>	<b>84</b>
<b>7</b>	<b>Discussion</b>	<b>85</b>
<b>8</b>	<b>Discussion</b>	<b>86</b>
<b>9</b>	<b>Discussion</b>	<b>87</b>
<b>10</b>	<b>Discussion</b>	<b>88</b>
<b>11</b>	<b>Discussion</b>	<b>89</b>
<b>12</b>	<b>Discussion</b>	<b>90</b>
<b>13</b>	<b>Discussion</b>	<b>91</b>
<b>14</b>	<b>Discussion</b>	<b>92</b>
<b>A</b>	<b>Form Penilaian Jurnal</b>	<b>93</b>
<b>B</b>	<b>FAQ</b>	<b>96</b>
	<b>Bibliography</b>	<b>98</b>

# List of Figures

1.1	Hasil Tampilan Error. . . . .	10
1.2	Hasil Tampilan Install joblib. . . . .	12
1.3	Hasil Tampilan Uji coba perintah joblib. . . . .	12
1.4	Download Anaconda. . . . .	12
1.5	Langkah pertama instalasi anaconda. . . . .	13
1.6	Langkah kedua instalasi anaconda. . . . .	13
1.7	Langkah ketiga instalasi anaconda. . . . .	14
1.8	Langkah terakhir instalasi anaconda. . . . .	14
1.9	Langkah pertama instalasi scikit pada CMD. . . . .	15
1.10	Langkah kedua instalasi scikit pada CMD. . . . .	15
1.11	Langkah ketiga instalasi scikit pada CMD. . . . .	15
1.12	Langkah compile code pada python anaconda. . . . .	16
1.13	Hasil Tampilan 1. . . . .	16
1.14	Hasil Tampilan 2. . . . .	16
1.15	Hasil Tampilan 3. . . . .	16
1.16	Hasil Tampilan 4. . . . .	16
1.17	Hasil Tampilan 5. . . . .	17
1.18	Hasil Tampilan 6. . . . .	17
1.19	Hasil Tampilan 7. . . . .	17
1.20	Hasil Tampilan 8. . . . .	17
1.21	Hasil Tampilan 9. . . . .	17
1.22	Hasil Tampilan 10. . . . .	18
1.23	Hasil Tampilan 11. . . . .	18
1.24	Hasil Tampilan 12. . . . .	18
1.25	Hasil Tampilan 13. . . . .	18
1.26	Hasil Tampilan 14. . . . .	18
1.27	Hasil Tampilan 15. . . . .	18
1.28	Hasil Tampilan 16. . . . .	18

1.29 Hasil Tampilan 17. . . . .	19
1.30 Hasil Tampilan 18. . . . .	19
1.31 Hasil Tampilan 19. . . . .	19
1.32 Hasil Tampilan 20. . . . .	19
1.33 Hasil Tampilan 21. . . . .	19
1.34 Hasil Tampilan 22. . . . .	19
2.1 Hasil Codingan No 1. . . . .	22
2.2 Hasil Codingan No 2. . . . .	23
2.3 Hasil Codingan No 3. . . . .	23
2.4 Hasil Codingan No 4. . . . .	24
2.5 Hasil Codingan No 5. . . . .	25
2.6 Hasil Codingan No 6. . . . .	25
2.7 Hasil Codingan No 7. . . . .	25
2.8 Hasil Codingan No 8. . . . .	26
2.9 Hasil Codingan No 9. . . . .	26
2.10 Hasil Codingan No 10. . . . .	27
2.11 Hasil Codingan No 11. . . . .	28
2.12 Hasil Codingan No 12. . . . .	29
2.13 Hasil Gambar Error No 6. . . . .	30
2.14 Hasil Gambar Penanganan Error No 6. . . . .	30
2.15 Binary Classification. . . . .	31
2.16 Supervised Learning. . . . .	32
2.17 Unsupervised Learning. . . . .	32
2.18 Clustering. . . . .	32
2.19 Evaluasi dan Akurasi. . . . .	33
2.20 K-fold Cross Validation. . . . .	33
2.21 Decision Tree. . . . .	34
2.22 Gain. . . . .	34
3.1 Random Forest . . . . .	42
3.2 Hasil Dataset . . . . .	43
3.3 Confusion Matrix . . . . .	44
3.4 Voting . . . . .	45
3.5 Aplikasi Menggunakan Pandas . . . . .	45
3.6 Aplikasi Menggunakan Numpy . . . . .	46
3.7 Aplikasi Menggunakan Matplotlib . . . . .	46



3.8	Hasil 1 Random Forest . . . . .	46
3.9	Hasil 2 Random Forest . . . . .	47
3.10	Hasil 3 Random Forest . . . . .	47
3.11	Hasil 4 Random Forest . . . . .	47
3.12	Hasil 5 Random Forest . . . . .	48
3.13	Hasil 6 Random Forest . . . . .	48
3.14	Hasil 7 Random Forest . . . . .	48
3.15	Hasil 8 Random Forest . . . . .	49
3.16	Hasil 9 Random Forest . . . . .	49
3.17	Hasil 10 Random Forest . . . . .	49
3.18	Hasil 11 Random Forest . . . . .	50
3.19	Hasil 11 Random Forest . . . . .	50
3.20	Hasil 12 Random Forest . . . . .	51
3.21	Hasil 13 Random Forest . . . . .	51
3.22	Hasil 14 Random Forest . . . . .	51
3.23	Hasil 15 Random Forest . . . . .	52
3.24	Hasil 16 Random Forest . . . . .	52
3.25	Hasil 17 Random Forest . . . . .	52
3.26	Hasil 1 Confusion Matrix . . . . .	52
3.27	Hasil 2 Confusion Matrix . . . . .	52
3.28	Hasil 3 Confusion Matrix . . . . .	53
3.29	Hasil 4 Confusion Matrix . . . . .	54
3.30	Hasil 5 Confusion Matrix . . . . .	55
3.31	Hasil 1 Klasifikasi SVM dan Decision Tree . . . . .	56
3.32	Hasil 2 Klasifikasi SVM dan Decision Tree . . . . .	56
3.33	Hasil 1 Cross Validation . . . . .	56
3.34	Hasil 2 Cross Validation . . . . .	56
3.35	Hasil 3 Cross Validation . . . . .	57
3.36	Hasil 1 Pengamatan Komponen Informasi . . . . .	57
3.37	Hasil 2 Pengamatan Komponen Informasi . . . . .	58
3.38	Error . . . . .	58
4.1	Klasifikasi Teks . . . . .	60
4.2	Klasifikasi Bunga . . . . .	60
4.3	Comment Youtube . . . . .	61
4.4	Bag Of Words . . . . .	62

4.5	TF-IDF . . . . .	63
4.6	Data Dummy 500 Data . . . . .	64
4.7	Membagi 2 Dataframe . . . . .	64
4.8	Vektorisasi dan Klasifikasi Data . . . . .	65
4.9	Data Content . . . . .	65
4.10	DataFrame Kata-kata Pada Content . . . . .	66
4.11	Klasifikasi SVM Dari Data Vektorisasi . . . . .	66
4.12	Klasifikasi Decision Tree Dari Data Vektorisasi . . . . .	67
4.13	Plot Confusion Matrix Menggunakan Matplotlib . . . . .	67
4.14	Program Cross Validation Pada Data Vektorisasi . . . . .	67
4.15	Program Pengamatan Komponen Informasi . . . . .	68
4.16	Error Pada Coding SVM . . . . .	68
5.1	Ilustrasi Soal No. 1 . . . . .	69
5.2	Ilustrasi Soal No. 2 . . . . .	70
5.3	Ilustrasi Soal No. 3 . . . . .	71
5.4	Ilustrasi Soal No. 4 . . . . .	71
5.5	Ilustrasi Soal No. 5 . . . . .	72
5.6	Ilustrasi Soal No. 6 . . . . .	72
5.7	Love . . . . .	73
5.8	Faith . . . . .	73
5.9	Fall . . . . .	73
5.10	Sick . . . . .	74
5.11	Clear . . . . .	74
5.12	Shine . . . . .	74
5.13	Bag . . . . .	74
5.14	Car . . . . .	75
5.15	Wash . . . . .	75
5.16	Motor . . . . .	75
5.17	Cycle . . . . .	76
5.18	Similariti Pada Kata Motor dan Cycle . . . . .	76
5.19	Similariti Pada Kata Wash dan Motor . . . . .	76
5.20	Extract_Words . . . . .	77
5.21	Permute Sentences . . . . .	77
5.22	Gensim TaggedDocument dan Doc2vec . . . . .	78
5.23	Aclimdb . . . . .	79

5.24	Aclimbdb . . . . .	79
5.25	Aclimbdb . . . . .	80
5.26	Infer Code . . . . .	81
5.27	Cosine Similarity . . . . .	82
5.28	Cross Validation Metode 1 . . . . .	83
5.29	Cross Validation Metode 2 . . . . .	83
5.30	Cross Validation Metode 3 . . . . .	83
A.1	Form nilai bagian 1. . . . .	94
A.2	form nilai bagian 2. . . . .	95

# Chapter 1

## Mengenai Kecerdasan Buatan dan Scikit-Learn

Buku umum yang digunakan adalah [3] dan untuk sebelum UTS menggunakan buku *Python Artificial Intelligence Projects for Beginners*[2]. Dengan praktek menggunakan python 3 dan editor anaconda dan library python scikit-learn. Tujuan pembelajaran pada pertemuan pertama antara lain:

1. Mengerti definisi kecerdasan buatan, sejarah kecerdasan buatan, perkembangan dan penggunaan di perusahaan
2. Memahami cara instalasi dan pemakaian sci-kit learn
3. Memahami cara penggunaan variabel explorer di spyder

Tugas dengan cara dikumpulkan dengan pull request ke github dengan menggunakan latex pada repo yang dibuat oleh asisten riset.

### 1.1 Teori

Praktek teori penunjang yang dikerjakan :

1. Buat Resume Definisi, Sejarah dan perkembangan Kecerdasan Buatan, dengan bahasa yang mudah dipahami dan dimengerti. Buatan sendiri bebas plagiat[hari ke 1](10)
2. Buat Resume mengenai definisi supervised learning, klasifikasi, regresi dan unsupervised learning. Data set, training set dan testing set.[hari ke 1](10)

## 1.2 Instalasi

Membuka <https://scikit-learn.org/stable/tutorial/basic/tutorial.html>. Dengan menggunakan bahasa yang mudah dimengerti dan bebas plagiat. Dan wajib skrinsut dari komputer sendiri.

1. Instalasi library scikit dari anaconda, mencoba kompilasi dan uji coba ambil contoh kode dan lihat variabel explorer[hari ke 1](10)
2. Mencoba Loading an example dataset, menjelaskan maksud dari tulisan tersebut dan mengartikan per baris[hari ke 1](10)
3. Mencoba Learning and predicting, menjelaskan maksud dari tulisan tersebut dan mengartikan per baris[hari ke 2](10)
4. mencoba Model persistence, menjelaskan maksud dari tulisan tersebut dan mengartikan per baris[hari ke 2](10)
5. Mencoba Conventions, menjelaskan maksud dari tulisan tersebut dan mengartikan per baris[hari ke 2](10)

## 1.3 Penanganan Error

Dari percobaan yang dilakukan di atas, apabila mendapatkan error maka:

1. skrinsut error[hari ke 2](10)
2. Tuliskan kode eror dan jenis errornya [hari ke 2](10)
3. Solusi pemecahan masalah error tersebut[hari ke 2](10)

## 1.4 Ahmad Syafrizal Huda/1164062

### 1.4.1 Teori

1. Definisi, sejarah, dan perkembangan kecerdasan buatan.

Definisi kecerdasan buatan adalah suatu pengetahuan yang dapat membuat komputer untuk meniru kecerdasan manusia yang berhubungan dengan penangkapan, pemodelan, dan penyimpanan kecerdasan manusia dalam sebuah sistem teknologi. Contohnya yaitu melakukan analisa penalaran untuk

mengambil suatu kesimpulan atau penerjemahan atau keputusan dari satu bahasa satu ke bahasa lain.

Sejarah dan perkembangan kecerdasan buatan terjadi pada musim panas tahun 1956 tercatat adanya seminar mengenai AI di Darmouth College. Seminar pada waktu itu dihadiri oleh sejumlah pakar komputer dan membahas potensi komputer dalam meniru kepandaian manusia. Akan tetapi perkembangan yang sering terjadi semenjak diciptakannya LISP, yaitu bahasa kecerdasan buatan yang dibuat tahun 1960 oleh John McCarthy. Istilah pada kecerdasan buatan atau Artificial Intelligence diambil dari Marvin Minsky dari MIT. Dia menulis karya ilmiah berjudul Step towards Artificial Intelligence, The Institute of radio Engineers Proceedings 49, January 1961[1].

2. Definisi supervised learning, klasifikasi, regresi, dan unsupervised learning. Data set, training set dan testing set.

Supervised learning merupakan sebuah pendekatan dimana sudah terdapat data yang dilatih, dan terdapat variable yang ditargetkan sehingga tujuan dari pendekatan ini adalah mengelompokkan suatu data ke data yang sudah ada. Sedangkan unsupervised learning tidak memiliki data latih, sehingga dari data yang ada, kita mengelompokkan data tersebut menjadi 2 bagian atau 3 bagian dan seterusnya.

Klasifikasi adalah salah satu topik utama dalam data mining atau machine learning. Klasifikasi yaitu suatu pengelompokan data dimana data yang digunakan tersebut mempunyai kelas label atau target.

Regresi adalah Supervised learning tidak hanya mempelajari classifier, tetapi juga mempelajari fungsi yang dapat memprediksi suatu nilai numerik. Contoh, ketika diberi foto seseorang, kita ingin memprediksi umur, tinggi, dan berat orang yang ada pada foto tersebut.

Data set adalah cabang aplikasi dari Artificial Intelligence/Kecerdasan Buatan yang fokus pada pengembangan sebuah sistem yang mampu belajar sendiri tanpa harus berulang kali di program oleh manusia.

Training set yaitu jika pasangan objek, dan kelas yang menunjuk pada objek tersebut adalah suatu contoh yang telah diberi label akan menghasilkan suatu algoritma pembelajaran.

Testing set digunakan untuk mengukur sejauh mana classifier berhasil melakukan klasifikasi dengan benar[4].

## 1.4.2 Instalasi

### 1.4.2.1 Instalasi Library Scikit dari Anaconda

1. Download aplikasi Anaconda terlebih dahulu. Lihat pada gambar 1.4.
2. Install aplikasi Anaconda yang sudah di download tadi. Lihat pada gambar 1.5.
3. Simpan aplikasi sesuai folder yang kita pilih lalu next. Lihat pada gambar 1.6.
4. Centang Keduanya lalu tekan tombol install. Lihat pada gambar 1.7.
5. Setelah itu tunggu sampai proses instalasi selesai lalu jika sudah tekan tombol finish. Lihat pada gambar 1.8.
6. Lalu buka command prompt anda dan tuliskan perintah berikut ini untuk mengecek apakah aplikasinya sudah terinstall. Lihat pada gambar 1.9.
7. Kemudian ketikkan perintah `pip install -U scikit-learn` seperti gambar berikut. Lihat pada gambar 1.10.
8. Lalu jika sudah ketikkan juga perintah `conda install scikit-learn`. Lihat pada gambar 1.11.
9. Hasil compile dari beberapa code yang mempunyai variable explorer. Lihat pada gambar 1.12.

### 1.4.2.2 Mencoba Loading an example Dataset

- `from sklearn import datasets`

(pada baris ini merupakan sebuah perintah untuk mengimport class datasets dari packaged sklearn).

- `iris = datasets.load_iris()`

(pada baris kedua ini dimana iris merupakan suatu estimator/parameter yang berfungsi untuk mengambil data pada item datasets.load\_iris).

- `digits = datasets.load_digits()`

(pada baris ketiga ini dimana digits merupakan suatu estimator/parameter yang berfungsi untuk mengambil data pada item datasets.load\_digits).

- `print(digits.data)`

(pada baris keempat ini merupakan perintah yang berfungsi untuk menampilkan estimator/parameter yang dipanggil pada item `digits.data` dan menampilkan outputannya) Lihat gambar 1.13.

- `digits.target`

(barisan ini untuk mengambil target pada estimator/parameter `digits` dan menampilkan outputannya) Lihat gambar 1.14.

- `digits.images[0]`

(barisan ini untuk mengambil `images[0]` pada estimator/parameter `digits` dan menampilkan outputannya) Lihat gambar 1.15.

#### 1.4.2.3 Learning and Predicting

- `from sklearn import svm`

(pada baris ini merupakan sebuah perintah untuk mengimport class `svm` dari packaged `sklearn`).

- `clf = svm.SVC(gamma=0.001, C=100.)`

(pada baris kedua ini `clf` sebagai estimator/parameter, `svm.SVC` sebagai class, `gamma` sebagai parameter untuk menetapkan nilai secara manual).

- `clf.fit(digits.data[:-1], digits.target[:-1])`

(pada baris ketiga ini `clf` sebagai estimator/parameter, `fit` sebagai metode, `digits.data` sebagai item, `[:-1]` sebagai syntax pythonnya dan menampilkan outputannya) Lihat gambar 1.16.

- `clf.predict(digits.data[-1:])`

(pada baris terakhir ini `clf` sebagai estimator/parameter, `predict` sebagai metode lainnya, `digits.data` sebagai item dan menampilkan outputannya) Lihat gambar 1.17.

#### 1.4.2.4 Model Persistence

- `from sklearn import svm`

(pada baris ini merupakan sebuah perintah untuk mengimport class `svm` dari packaged `sklearn`).



- `from sklearn import datasets`

(pada baris ini merupakan sebuah perintah untuk mengimport class datasets dari packaged sklearn).

- `clf = svm.SVC(gamma='scale')`

(pada baris ketiga ini clf sebagai estimator/parameter, svm.SVC sebagai class, gamma sebagai parameter untuk menetapkan nilai secara manual dengan nilai scale).

- `iris = datasets.load_iris()`

(pada baris keempat ini iris sebagai estimator/parameter, datasets.load\_iris() sebagai item dari suatu nilai).

- `X, y = iris.data, iris.target`

(pada baris kelima ini X, y sebagai estimator/parameter, iris.data, iris.target sebagai item dari 2 nilai yang ada).

- `clf.fit(X, y)`

(pada baris keenam ini clf sebagai estimator/parameter dengan menggunakan metode fit untuk memanggil estimator X, y dengan outputannya) Lihat gambar 1.18.

- `import pickle`

(pickle merupakan sebuah class yang di import).

- `s = pickle.dumps(clf)`

(pada baris ini s sebagai estimator/parameter dengan pickle.dumps merupakan suatu nilai/item dari estimator/parameter clf)

- `clf2 = pickle.loads(s)`

(pada baris ini clf2 sebagai estimator/parameter, pickle.loads sebagai suatu item, dan s sebagai estimator/parameter yang dipanggil)

- `clf2.predict(X[0:1])`

(pada baris ini clf2.predict sebagai suatu item dengan menggunakan metode predict untuk menentukan suatu nilai dari (X[0:1])) Lihat gambar 1.19.

- `y[0]`

(pada estimator/parameter `y` berapapun angka yang diganti nilainya akan selalu konstan yaitu 0) Lihat gambar 1.20.

- `from joblib import dump, load`

(pada baris berikut ini merupakan sebuah perintah untuk mengimport class `dump`, `load` dari packaged `joblib`).

- `dump(clf, 'filename.joblib')`

(pada baris berikutnya `dump` di sini sebagai class yang didalamnya terdapat nilai dari suatu item `clf` dan data `joblib`).

- `clf = load('filename.joblib')`

(pada baris terakhir `clf` sebagai estimator/parameter dengan suatu nilai `load` berfungsi untuk mengulang data sebelumnya)

- dari ketiga baris akhir tersebut jika di jalankan aau dituliskan perintah seperti itu maka akan menampilkan tampilan eror terlihat pada gambar 1.21.

#### 1.4.2.5 Conventions

##### 1. Type Casting

- `from sklearn import svm`

(pada baris ini merupakan sebuah perintah untuk mengimport class `svm` dari packaged `sklearn`).

- `from sklearn import random_projection`

(pada baris ini merupakan sebuah perintah untuk mengimport class `random_projection` dari packaged `sklearn`).

- `rng = np.random.RandomState(0)`

(`rng` sebagai estimator/parameter dengan nilai suatu itemnya yaitu `np.random.RandomS`)

- `X = rng.rand(10, 2000)`

(`X` sebagai estimator/parameter dengan nilai item `rng.rand`).

- `X = np.array(X, dtype='float32')`

(`X` sebagai estimator/parameter dengan nilai item `np.array`).

- `X.dtype`  
(`X.dtype` sebagai item pemanggil) Lihat gambar 1.22.
- `transformer = random_projection.GaussianRandomProjection()`  
(`transformer` sebagai estimator/parameter dengan memanggil class `random_projection`).
- `X_new = transformer.fit_transform(X)`  
(`X_new` di sini sebagai estimator/parameter dan menggunakan metode `fit`)
- `X_new.dtype`  
(`X_new.dtype` sebagai item) Lihat gambar 1.23.
- `from sklearn import datasets`  
(pada baris ini merupakan sebuah perintah untuk mengimport class `datasets` dari packaged `sklearn`).
- `from sklearn.svm import SVC`  
(pada baris ini merupakan sebuah perintah untuk mengimport class `SVC` dari packaged `sklearn.svm`).
- `iris = datasets.load_iris()`  
(`iris` sebagai estimator/parameter dengan item `datasets.load_iris()`).
- `clf = SVC(gamma='scale')`  
(`clf` sebagai estimator/parameter dengan nilai class `SVC` pada parameter `gamma` sebagai set penilaian).
- `clf.fit(iris.data, iris.target)`  
(estimator/parameter `clf` menggunakan metode `fit` dengan itemnya) Lihat gambar 1.24.
- `list(clf.predict(iris.data[:3]))`  
(menambahkan item `list` dengan metode `predict`) Lihat gambar 1.25.
- `clf.fit(iris.data, iris.target_names[iris.target])`  
(estimator/parameter `clf` menggunakan metode `fit` dengan itemnya) Lihat gambar 1.26.
- `list(clf.predict(iris.data[:3]))` (menambahkan item `list` dengan metode `predict`)  
Lihat gambar 1.27.

## 2. Refitting and Updating Parameters

- `import numpy as np`  
(pada baris ini merupakan sebuah perintah untuk mengimport class svm dari np).
- `from sklearn.svm import SVC`  
(pada baris ini merupakan sebuah perintah untuk mengimport class SVC dari packaged sklearn.svm).
- `rng = np.random.RandomState(0)`  
(rng sebagai estimator/parameter dengan nilai suatu itemnya yaitu np.random.RandomS
- `X = rng.rand(100, 10)`  
(X sebagai estimator/parameter dengan nilai item rng.rand).
- `y = rng.binomial(1, 0.5, 100)`  
(y sebagai estimator/parameter dengan nilai item rng.binomial).
- `X_test = rng.rand(5, 10)`  
(X\_test sebagai estimator/parameter dengan nilai item rng.rand).
- `clf = SVC()`  
(clf sebagai estimator/parameter dan class SVC)
- `clf.set_params(kernel='linear').fit(X, y)`  
(set\_params sebagai item) Lihat gambar 1.28.
- `clf.predict(X_test)`  
(menggunakan metode predict) Lihat gambar 1.29.
- `clf.set_params(kernel='rbf', gamma='scale').fit(X, y)`  
Lihat gambar 1.30.
- `clf.predict(X_test)`  
Lihat gambar 1.31.

### 3. Multiclass vs. Multilabel Fitting

- `from sklearn.svm import SVC`  
(pada baris ini merupakan sebuah perintah untuk mengimport class SVC dari packaged sklearn.svm).
- `from sklearn.multiclass import OneVsRestClassifier`  
(pada baris ini merupakan sebuah perintah untuk mengimport class OneVs-RestClassifier dari packaged sklearn.multiclass).

- `from sklearn.preprocessing import LabelBinarizer`  
(pada baris ini merupakan sebuah perintah untuk mengimport class LabelBinarizer dari packaged sklearn.preprocessing).
- `X = [[1, 2], [2, 4], [4, 5], [3, 2], [3, 1]]`
- `y = [0, 0, 1, 1, 2]`
- `classif = OneVsRestClassifier(estimator=SVC(gamma='scale', random_state=0))`
- `classif.fit(X, y).predict(X)`

Lihat gambar 1.32.

- `y = LabelBinarizer().fit_transform(y)`
- `classif.fit(X, y).predict(X)`

Lihat gambar 1.33.

- `from sklearn.preprocessing import MultiLabelBinarizer`
- `y = [[0, 1], [0, 2], [1, 3], [0, 2, 3], [2, 4]]`
- `y = MultiLabelBinarizer().fit_transform(y)`
- `classif.fit(X, y).predict(X)`

Lihat gambar 1.34.

### 1.4.3 Penanganan error

#### 1.4.3.1 ScreenShoot Error

```
>>> from joblib import dump, load
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'joblib'
>>> dump(clf, 'filename.joblib')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'dump' is not defined
>>> clf = load('filename.joblib')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'load' is not defined
```

Figure 1.1: Hasil Tampilan Error.

#### 1.4.3.2 Tuliskan Kode Error dan Jenis Erornya

- `from joblib import dump, load`

(Kode baris pertama)

```
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
ModuleNotFoundError: No module named 'joblib'
```

(Errornya)

- `dump(clf, 'filename.joblib')`

(Kode baris kedua)

```
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
NameError: name 'dump' is not defined
```

(Errornya)

- `clf = load('filename.joblib')`

(Kode baris ketiga)

```
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
NameError: name 'load' is not defined
```

(Errornya)

#### 1.4.3.3 Solusi Pemecahan Masalah Error

1. Pada masalah error sebelumnya itu dikarenakan kita belum mempunyai packaged joblib. Jadi solusinya yaitu dengan cara menginstall terlebih dahulu packaged joblibnya setelah itu baru perintah tersebut dapat dijalankan seperti pada gambar 1.2 dan 1.3

```
Command Prompt
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\HUDA>pip install joblib
Requirement already satisfied: joblib in f:\anaconda\lib\site-packages (0.13.2)
distributed 1.21.0 requires msgpack, which is not installed.
You are using pip version 10.0.1, however version 19.0.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\HUDA>
```

Figure 1.2: Hasil Tampilan Install joblib.

```
>>> from joblib import dump, load
>>> dump(clf, 'filename.joblib')
['filename.joblib']
>>> clf = load('filename.joblib')
>>>
```

Figure 1.3: Hasil Tampilan Uji coba perintah joblib.

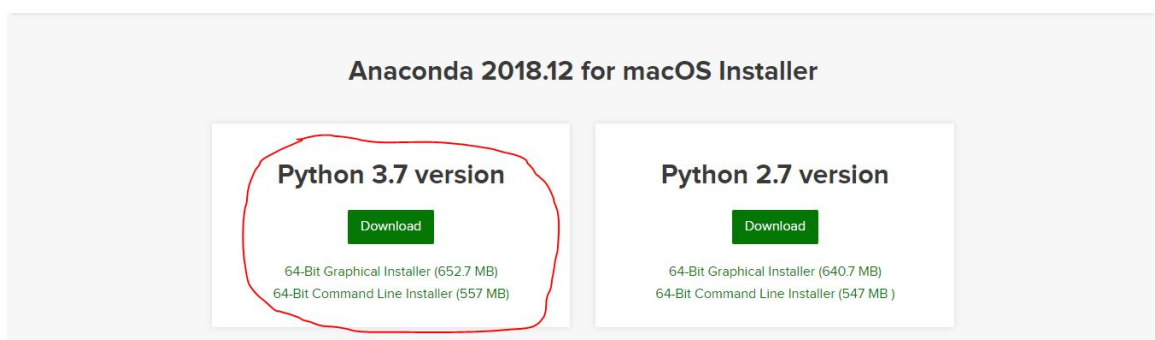


Figure 1.4: Download Anaconda.

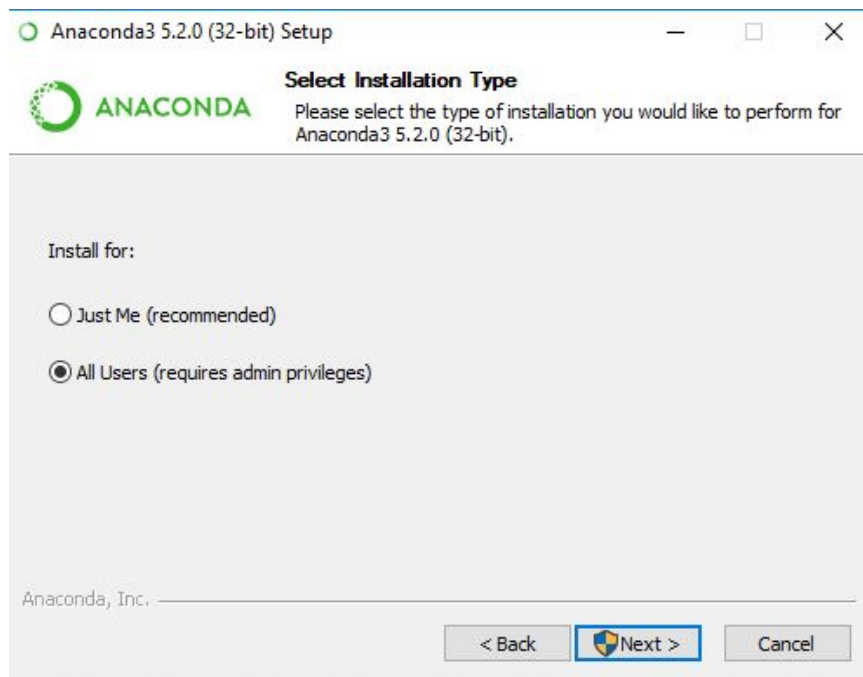


Figure 1.5: Langkah pertama instalasi anaconda.

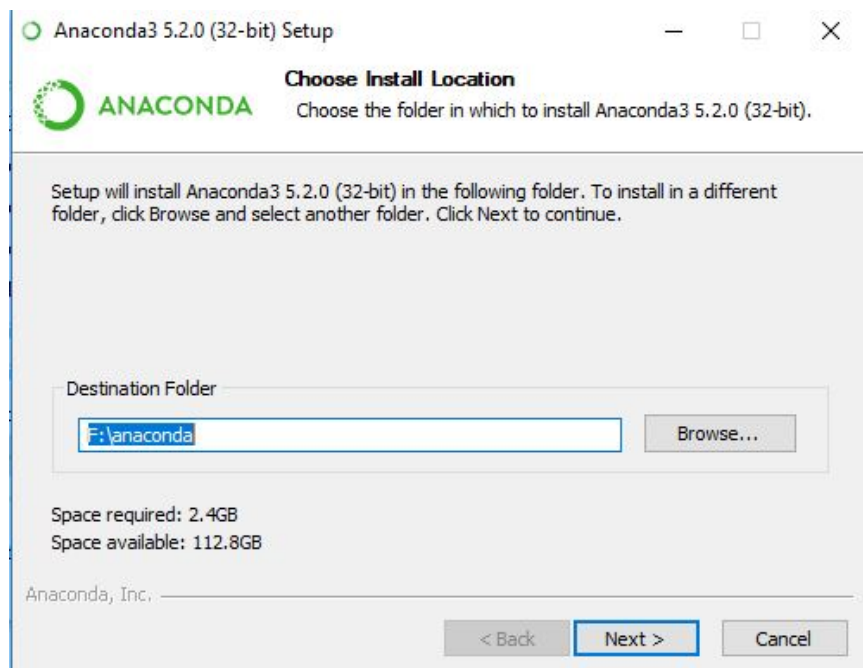


Figure 1.6: Langkah kedua instalasi anaconda.



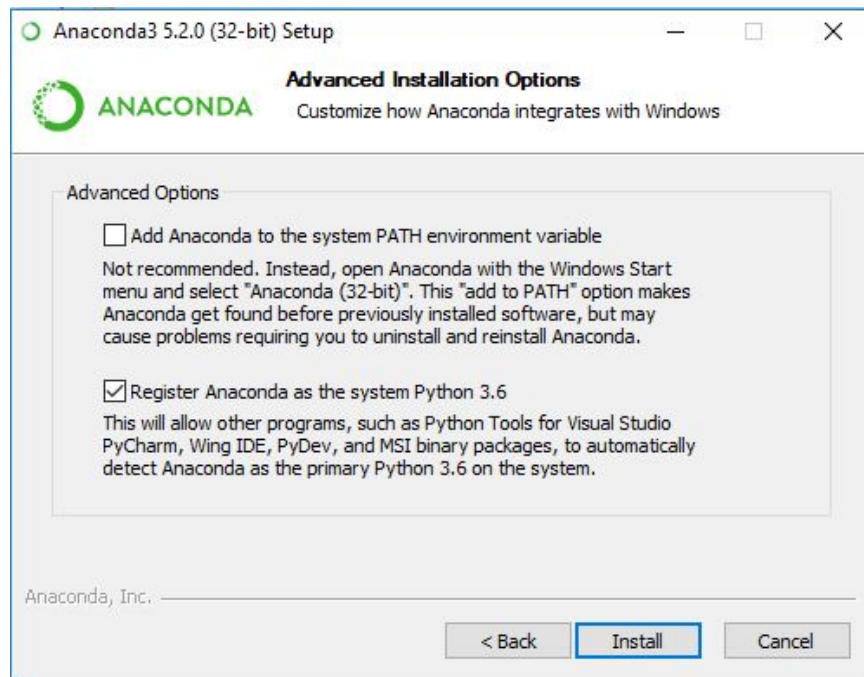


Figure 1.7: Langkah ketiga instalasi anaconda.

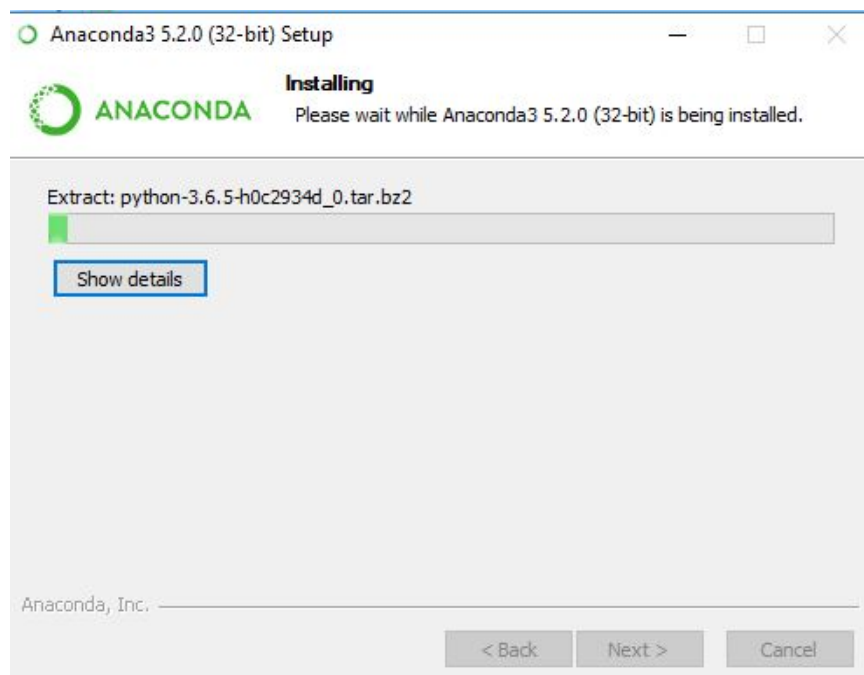


Figure 1.8: Langkah terakhir instalasi anaconda.

```
C:\ Command Prompt
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\HUDA>conda --version
conda 4.5.4

C:\Users\HUDA>python --version
Python 3.6.5 :: Anaconda, Inc.
```

Figure 1.9: Langkah pertama instalasi scikit pada CMD.

```
C:\Users\HUDA>pip install -U scikit-learn
Collecting scikit-learn
  Downloading https://files.pythonhosted.org/packages/ee/cb/c89ebdc0d7dbba6e6fd222daabd257da3c28a967dd7c352d4272b2e1cef6/scikit_learn-0.20.2-cp36-cp36m-win32.whl (4.3MB)
    100% |#####| 4.3MB 425kB/s
Requirement not upgraded as not directly required: numpy>=1.8.2 in f:\anaconda\lib\site-packages (from scikit-learn) (1.14.3)
Requirement not upgraded as not directly required: scipy>=0.13.3 in f:\anaconda\lib\site-packages (from scikit-learn) (1.1.0)
Installing collected packages: scikit-learn
  Found existing installation: scikit-learn 0.19.1
    Uninstalling scikit-learn-0.19.1:
      Successfully uninstalled scikit-learn-0.19.1
  Successfully installed scikit-learn-0.20.2
You are using pip version 10.0.1, however version 19.0.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

Figure 1.10: Langkah kedua instalasi scikit pada CMD.

```
C:\Users\HUDA>conda install scikit-learn
Solving environment: done

## Package Plan ##

  environment location: F:\anaconda

  added / updated specs:
    - scikit-learn

The following packages will be downloaded:



| package     | build  |        |
|-------------|--------|--------|
| conda-4.6.7 | py36_0 | 1.7 MB |



The following packages will be UPDATED:

  conda: 4.5.4-py36_0 --> 4.6.7-py36_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
conda-4.6.7 | 1.7 MB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Figure 1.11: Langkah ketiga instalasi scikit pada CMD.

```

C:\Users\HUDA>python
Python 3.6.5 [Anaconda, Inc.] (default, Mar 29 2018, 13:23:52) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from sklearn.metrics import confusion_matrix
>>> y_true = [2, 0, 2, 2, 0, 1]
>>> y_pred = [0, 0, 2, 2, 0, 2]
>>> confusion_matrix(y_true, y_pred)
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]], dtype=int64)
>>> y_true = ["cat", "ant", "cat", "cat", "ant", "bird"]
>>> y_pred = ["ant", "ant", "cat", "cat", "ant", "cat"]
>>> confusion_matrix(y_true, y_pred, labels=["ant", "bird", "cat"])
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]], dtype=int64)
>>> tn, fp, fn, tp = confusion_matrix([0, 1, 0, 1], [1, 1, 1, 0]).ravel()
>>> (tn, fp, fn, tp)
(0, 2, 1, 1)
>>>

```

Figure 1.12: Langkah compile code pada python anaconda.

```

[[ 0.  0.  5. ...  0.  0.  0.]
 [ 0.  0.  0. ... 10.  0.  0.]
 [ 0.  0.  0. ... 16.  9.  0.]
 ...
 [ 0.  0.  1. ...  6.  0.  0.]
 [ 0.  0.  2. ... 12.  0.  0.]
 [ 0.  0. 10. ... 12.  1.  0.]]

```

Figure 1.13: Hasil Tampilan 1.

```

array([0, 1, 2, ..., 8, 9, 8])

```

Figure 1.14: Hasil Tampilan 2.

```

array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])

```

Figure 1.15: Hasil Tampilan 3.

```

SVC(C=100.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)

```

Figure 1.16: Hasil Tampilan 4.

A terminal window with a black background and light blue text. The text displays 'array([8])'.

Figure 1.17: Hasil Tampilan 5.

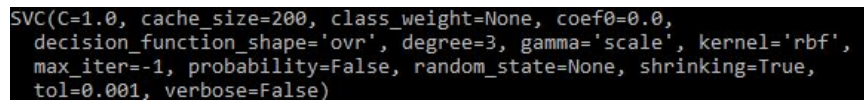
A terminal window with a black background and light blue text. The text displays the parameters of an SVC model: SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0, decision\_function\_shape='ovr', degree=3, gamma='scale', kernel='rbf', max\_iter=-1, probability=False, random\_state=None, shrinking=True, tol=0.001, verbose=False).

Figure 1.18: Hasil Tampilan 6.

A terminal window with a black background and light blue text. The text displays 'array([0])'.

Figure 1.19: Hasil Tampilan 7.

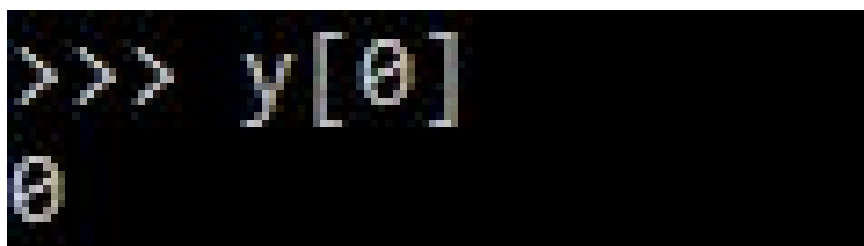
A terminal window with a black background and light blue text. The text displays a prompt '>>>' followed by 'y[0]' and the output '0'.

Figure 1.20: Hasil Tampilan 8.

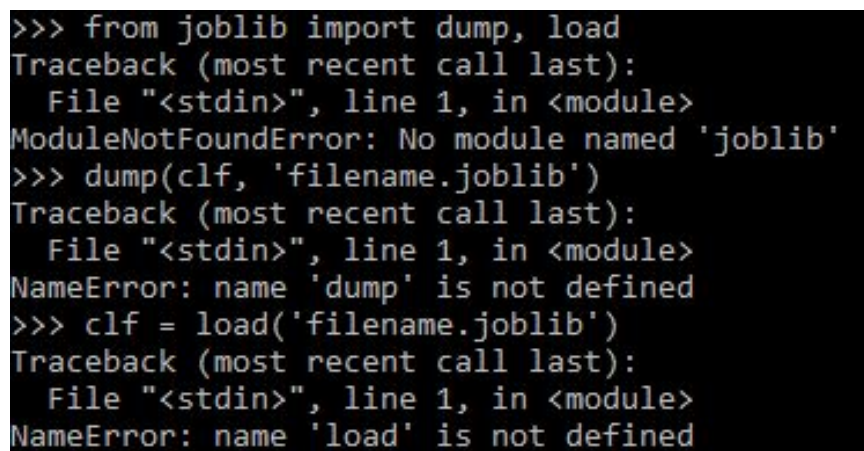
A terminal window with a black background and light blue text. The text displays a series of commands and errors: '>>> from joblib import dump, load' followed by a 'ModuleNotFoundError: No module named 'joblib'', '>>> dump(clf, 'filename.joblib')' followed by a 'NameError: name 'dump' is not defined', and '>>> clf = load('filename.joblib')' followed by a 'NameError: name 'load' is not defined'.

Figure 1.21: Hasil Tampilan 9.

```
>>> X.dtype
dtype('float32')
```

Figure 1.22: Hasil Tampilan 10.

```
>>> X_new.dtype
dtype('float64')
```

Figure 1.23: Hasil Tampilan 11.

```
>>> clf.fit(iris.data, iris.target)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Figure 1.24: Hasil Tampilan 12.

```
>>> list(clf.predict(iris.data[:3]))
[0, 0, 0]
```

Figure 1.25: Hasil Tampilan 13.

```
>>> clf.fit(iris.data, iris.target_names[iris.target])
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Figure 1.26: Hasil Tampilan 14.

```
>>> list(clf.predict(iris.data[:3]))
['setosa', 'setosa', 'setosa']
```

Figure 1.27: Hasil Tampilan 15.

```
>>> clf.set_params(kernel='linear').fit(X, y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

Figure 1.28: Hasil Tampilan 16.

```
>>> clf.predict(X_test)
array([1, 0, 1, 1, 0])
```

Figure 1.29: Hasil Tampilan 17.

```
>>> clf.set_params(kernel='rbf', gamma='scale').fit(X, y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Figure 1.30: Hasil Tampilan 18.

```
>>> clf.predict(X_test)
array([1, 0, 1, 1, 0])
```

Figure 1.31: Hasil Tampilan 19.

```
>>> classif.fit(X, y).predict(X)
array([0, 0, 1, 1, 2])
```

Figure 1.32: Hasil Tampilan 20.

```
>>> classif.fit(X, y).predict(X)
array([[1, 0, 0],
       [1, 0, 0],
       [0, 1, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

Figure 1.33: Hasil Tampilan 21.

```
>>> classif.fit(X, y).predict(X)
array([[1, 1, 0, 0, 0],
       [1, 0, 1, 0, 0],
       [0, 1, 0, 1, 0],
       [1, 0, 1, 0, 0],
       [1, 0, 1, 0, 0]])
```

Figure 1.34: Hasil Tampilan 22.

# Chapter 2

## Related Works

Your related works, and your purpose and contribution which must be different as below.

### 2.1 Ahmad Syafrizal Huda/1164062

#### 2.1.1 Teori

1. Binary Classification yaitu katakanlah kita memiliki tugas untuk mengklasifikasi objek menjadi dua kelompok berdasarkan beberapa fitur. Sebagai contoh, katakanlah kita diberi beberapa pena dan pensil dari berbagai jenis dan merek, kita dapat dengan mudah memisahnya menjadi dua kelas, yaitu pena dan pensil.

Contoh ilustrasi gambar bisa dilihat pada gambar 2.15.

2. Supervised learning merupakan sebuah pendekatan dimana sudah terdapat data yang dilatih, dan terdapat variable yang ditargetkan sehingga tujuan dari pendekatan ini adalah mengelompokkan suatu data ke data yang sudah ada. Sedangkan unsupervised learning tidak memiliki data latih, sehingga dari data yang ada, kita mengelompokkan data tersebut menjadi 2 bagian atau 3 bagian dan seterusnya. Dan clustering adalah proses pengelompokan entitas yang sama bersama-sama. Tujuan dari teknik pembelajaran mesin tanpa pengawasan ini adalah untuk menemukan kesamaan pada titik data dan mengelompokkan titik data yang serupa secara bersamaan[4].

Contoh ilustrasi gambar bisa dilihat pada gambar 2.16.

Contoh ilustrasi gambar bisa dilihat pada gambar 2.17.

Contoh ilustrasi gambar bisa dilihat pada gambar 2.18.

3. Evaluasi dan akurasi adalah bagaimana cara kita bisa mengevaluasi seberapa baik model mengerjakan pekerjaannya dengan cara mengukur akurasinya. Akurasi nantinya didefinisikan sebagai presentase kasus yang telah diklasifikasikan dengan benar. Kita dapat melakukan analisis kesalahan yang telah di buat oleh model.

Contoh ilustrasi gambar bisa dilihat pada gambar 2.19.

4. Cara membuat dan membaca confusion matrix yaitu, menentukan pokok permasalahan serta atributnya, membuat Decision Tree, membuat Data Testing, mencari nilai variabelnya misal a,b,c, dan d, mencari nilai recall, precision, accuracy, dan error rate.

Contoh Confusion Matrix.

$$\text{Recall} = 3/(1+3) = 0,75$$

$$\text{Precision} = 3/(1+3) = 0,75$$

$$\text{Accuracy} = (5+3)/(5+1+1+3) = 0,8$$

$$\text{Error Rate} = (1+1)/(5+1+1+3) = 0,2$$

5. Berikut ini tata cara kerja K-fold Cross Validation;

- Total instance akan dibagi menjadi N bagian.
- Fold yang pertama adalah bagian pertama menjadi testing data dan sisanya menjadi training data.
- Hitung akurasi berdasarkan porsi data tersebut dengan menggunakan persamaan.
- Fold yang ke dua adalah bagian ke dua menjadi testing data dan sisanya training data.
- Hitung akurasi berdasarkan porsi data tersebut.
- Lakukan step secara berulang hingga habis mencapai fold ke-K.
- Terakhir hitung rata-rata akurasi K buah.

Contoh ilustrasi gambar bisa dilihat pada gambar 2.20.

6. Decision Tree adalah sebuah metode pembelajaran yang digunakan untuk melakukan klarifikasi dan regresi. Decision Tree digunakan untuk membuat sebuah model



yang dapat memprediksi sebuah nilai variabel target dengan cara mempelajari aturan keputusan dari fitur data. Contoh Decision Tree adalah untuk melakukan prediksi apakah Kuda termasuk hewan mamalia atau bukan.

Contoh ilustrasi gambar bisa dilihat pada gambar 2.21.

7. Gain adalah pengurangan yang diharapkan dalam entropy. Dalam machine learning, gain dapat digunakan untuk menentukan sebuah urutan atribut atau memperkecil atribut yang telah dipilih. Urutan ini akan membentuk decision tree, atribut gain dipilih yang paling besar. Dan Entropi adalah ukuran ketidakpastian sebuah variabel acak sehingga dapat diartikan entropi adalah ukuran ketidakpastian dari sebuah atribut.

Contoh ilustrasi gambar bisa dilihat pada gambar 2.22.

### 2.1.2 Scikit-learn

1. Penjelasan Codingan ini akan menampilkan data pada file yang ditentukan. Untuk codingan ini file yang dieksekusi untuk digunakan ialah student-mat.csv. Secara jelasnya, dalam codingan dapat dilihat bahwa variabel buahpir didefinisikan untuk pembacaan csv dari "buahnaga" dimana untuk pemisahannya yaitu separation berupa ; . Setelah itu variabel buahpir di tampilkan dengan perintah menampilkan len panjang ataupun jumlah dan hasilnya berupa angka 395 .

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.1.

```
In [10]: import pandas as buahnaga
...: buahpir = buahnaga.read_csv('E:\DATA KULIAH\SEMESTER 6\KECERDASAN BUATAN(PAK
ROLLY)\Python-Artificial-Intelligence-Projects-for-Beginners\Chapter01\dataset\student-
mat.csv', sep=';')
...: len(buahpir)
Out[10]: 395
```

Figure 2.1: Hasil Codingan No 1.

2. Penjelasan codingan ini berfungsi untuk menampilkan baris G1, G2 dan G3 ( berdasarkan kriterianya ) untuk kolom PASS pada variabel buahpir. Untuk lebih jelasnya, pada codingan terdapat pendefinisian pembacaan lamda ( panjang gelombang ) dari baris G1, G2 dan G3. Apabila row-row tersebut bernilai lebih dari 35 maka akan terdefiniskan angka 1 apabila tidak, maka akan terdefiniskan angka 0 pada kolom PASS ( sesuai permintaan awal ). Selanjutnya

variabelnya di ditampilkan sehingga menampilkan keluaran. Tidak lupa terdapat juga jumlah dari baris dan kolom yang berubah sesuai dengan baris yang dieksekusi.

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.2.

```
In [11]: buahpir['pass'] = buahpir.apply(lambda row: 1 if (row['G1']+row['G2']+row['G3'])
...: >= 35 else 0, axis=1)
...: buahpir = buahpir.drop(['G1', 'G2', 'G3'], axis=1)
...: buahpir.head()
Out[11]:
```

	school	sex	age	address	famsize	...	Dalc	Walc	health	absences	pass
0	GP	F	18	U	GT3	...	1	1	3	6	0
1	GP	F	17	U	GT3	...	1	1	3	4	0
2	GP	F	15	U	LE3	...	2	3	3	10	0
3	GP	F	15	U	GT3	...	1	1	5	2	1
4	GP	F	16	U	GT3	...	1	2	5	4	0

[5 rows x 31 columns]

Figure 2.2: Hasil Codingan No 2.

3. Penjelasan codingan ini mendefinisikan pemanggilan get dummies dari buahnaga dalam variabel buahpir. Di dalam get dummies sendiri akan terdefiniskan variabel buahpir dengan kolom-kolom yang akan dieksekusi seperti school, address dll. Kemudian variabel tersebut diartikan untuk mendapatkan kembalian berupa keluaran dari eksekusi perintah variabel buahpir beserta dengan jumlah baris dan kolom data yang dieksekusi.

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.3.

```
In [12]: buahpir = buahnaga.get_dummies(buahpir, columns=['sex', 'school', 'address',
...: 'famsize',
...: 'Pstatus', 'Mjob', 'Fjob',
...: 'reason', 'guardian', 'schoolsup',
...: 'famsup', 'paid', 'activities',
...: 'nursery', 'higher', 'internet',
...: 'romantic'])
...: buahpir.head()
Out[12]:
```

	age	Medu	Fedu	...	internet_yes	romantic_no	romantic_yes
0	18	4	4	...	0	1	0
1	17	1	1	...	1	1	0
2	15	1	1	...	1	1	0
3	15	4	2	...	1	0	1
4	16	3	3	...	0	1	0

[5 rows x 57 columns]

Figure 2.3: Hasil Codingan No 3.

4. Penjelasan codingan ini difungsikan untuk mengartikan pembagian data yang berupa training dan testing data. pertama-tama variabel buahpir akan mengartikan sampel yang akan digunakan ( berupa shuffle row ) . Nah kemudian masing-masing parameter yaitu buahpir train dan buahpir test akan berjumlah 500 data ( telah dibagi untuk training dan testing ). Selanjutnya dilakukan pengekseskuan untuk kolom Pass, apabila sesuai dengan axis=1 maka eksekusi fungsi berhasil. Selain itu juga disertakan jumlah dari peserta yang lolos dari semua nilai data setnya.

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.4.

```
In [5]: buahpir = buahpir.sample(frac=1)
...: # split training and testing data
...: buahpir_train = buahpir[:500]
...: buahpir_test = buahpir[500:]
...:
...: buahpir_train_att = buahpir_train.drop(['pass'], axis=1)
...: buahpir_train_pass = buahpir_train['pass']
...:
...: buahpir_test_att = buahpir_test.drop(['pass'], axis=1)
...: buahpir_test_pass = buahpir_test['pass']
...:
...: buahpir_att = buahpir.drop(['pass'], axis=1)
...: buahpir_pass = buahpir['pass']
...:
...: # number of passing students in whole dataset:
...: import numpy as buahpepaya
...: print("Passing: %d out of %d (%.2f%%)" % (buahpepaya.sum(buahpir_pass),
len(buahpir_pass), 100*float(buahpepaya.sum(buahpir_pass)) / len(buahpir_pass)))
Passing: 328 out of 649 (50.54%)
```

Figure 2.4: Hasil Codingan No 4.

5. Penjelasan codingan ini hanya membuktikan pengujian dari Klasifikasi Decision Tree yang ada, apakah true atau tidak dan hasilnya true. Pada codingan ini di definisikan library sklearn untuk mengimpor atau menampilkan tree. Variabel buahapel difungsikan untuk membaca klasifikasi decision tree dari tree itu sendiri dengan 2 parameternya yaitu kriteria=entropy dan max depth=5. Maka selanjutnya variabel buahapel akan masuk dan terbaca dalam module fit dengan 2 parameter yaitu buahpir trai att dan buahpir train pass.

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.5.

6. Penjelasan codingan ini memberikan gambaran dari klasifikasi decision tree

```
In [6]: from sklearn import tree
...: buahapel = tree.DecisionTreeClassifier(criterion="entropy", max_depth=5)
...: buahapel = buahapel.fit(buahpir_train_att, buahpir_train_pass)
```

Figure 2.5: Hasil Codingan No 5.

yaitu pengolahan parameter yang dieksekusi kedalam variabel buahapel. Tentunya dengan pemanfaatan library graphviz yang telah diimport dan difungsikan.

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.6.

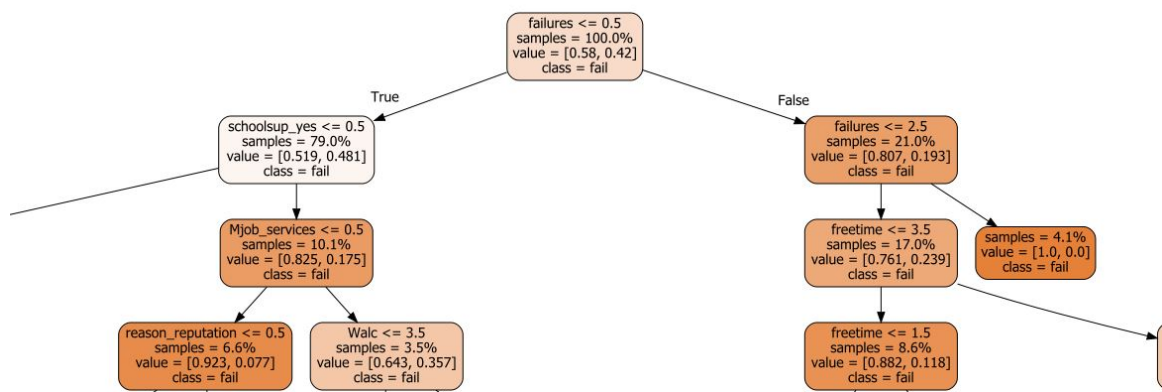


Figure 2.6: Hasil Codingan No 6.

- Penjelasan codingan ini membahas tentang penyimpanan tree dari library graphviz yang dieksekusi bersamaan dengan variabel buahapel dan parameter lainnya. Dilakukan pengecekan dan pengujian apakah klasifikasi decision treenya dapat berjalan atau tidak. Apabila tidak berjalan, maka akan terjadi error, namun codingan ini berfungsi.

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.7.

```
In [7]: tree.export_graphviz(buahapel, out_file="student-performance.dot",
label="all", impurity=False, proportion=True,
...: feature_names=list(buahpir_train_att),
class_names=["fail", "pass"],
...: filled=True, rounded=True)
```

Figure 2.7: Hasil Codingan No 7.

- Penjelasan codingan ini membaca score dari variabel buahapel dimana terdapat 2 parameter yang dihitung dan diuji yaitu buahpir test att dan buahpir test pass.

Untuk hasilnya sendiri mengapa berupa angka, dikarenakan pada parameter yang dieksekusi memang memiliki data sehingga dieksekusi dan menghasilkan keluaran dari score tersebut.

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.8.

```
In [8]: buahapel.score(buahpir_test_att, buahpir_test_pass)
Out[8]: 0.6644295302013423
```

Figure 2.8: Hasil Codingan No 8.

9. Penjelasan codingan ini membahas mengenai pengkesekusian fungsi dan variabel dari library yang didefinisikan dan yang diimport. Penjelasan lebih jelasnya ialah codingan ini mendefinisikan library sklearn.model.selection kemudian mengimport cross val score. Kemudian variabel score mendefinisikan cross val score yang telah diimport tadi dengan 4 parameter yaitu buahapel, buahpir att, buahpir pass dan cv=5 untuk dieksekusi. Setelah semua pemrosesan tersebut maka hasil yang di tampilkan ialah rata2 perhitungan dari variabel score dimana dan standar dari plus minusnya tentunya dengan ketentuan parameter Accuracy .

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.9.

```
In [8]: from sklearn.model_selection import cross_val_score
...: scores = cross_val_score(buahapel, buahpir_att, buahpir_pass, cv=5)
...: # show average score and +/- two standard deviations away (covering 95% of
scores)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
Accuracy: 0.68 (+/- 0.07)
```

Figure 2.9: Hasil Codingan No 9.

10. Penjelasan Codingan ini mendefinisikan max depth dalam jarak angka antara parameter 1 dan 20. Variabel buahapel mendefinisikan klasifikasi decision tree dengan 2 parameter. Kemudian variabel score mengeksekusi parameter lainnya yaitu seperti buahapel, buahpir att, buahpir pass dan cv=5 ) . Hasil yang ditampilkan ialah dari max depth, accuracy dan plus minusnya dan akhirnya hasil outputannya keluar.

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.10.



```

In [10]: for max_depth in range(1, 20):
...:     buahapel = tree.DecisionTreeClassifier(criterion="entropy", max_depth=max_depth)
...:     scores = cross_val_score(buahapel, buahpir_att, buahpir_pass, cv=5)
...:     print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" % (max_depth, scores.mean(),
scores.std() * 2))
Max depth: 1, Accuracy: 0.64 (+/- 0.02)
Max depth: 2, Accuracy: 0.69 (+/- 0.02)
Max depth: 3, Accuracy: 0.69 (+/- 0.07)
Max depth: 4, Accuracy: 0.69 (+/- 0.05)
Max depth: 5, Accuracy: 0.67 (+/- 0.03)
Max depth: 6, Accuracy: 0.66 (+/- 0.08)
Max depth: 7, Accuracy: 0.67 (+/- 0.07)
Max depth: 8, Accuracy: 0.67 (+/- 0.05)
Max depth: 9, Accuracy: 0.65 (+/- 0.04)
Max depth: 10, Accuracy: 0.66 (+/- 0.05)
Max depth: 11, Accuracy: 0.66 (+/- 0.07)
Max depth: 12, Accuracy: 0.62 (+/- 0.09)
Max depth: 13, Accuracy: 0.64 (+/- 0.08)
Max depth: 14, Accuracy: 0.63 (+/- 0.09)
Max depth: 15, Accuracy: 0.64 (+/- 0.07)
Max depth: 16, Accuracy: 0.63 (+/- 0.07)
Max depth: 17, Accuracy: 0.62 (+/- 0.09)
Max depth: 18, Accuracy: 0.63 (+/- 0.08)
Max depth: 19, Accuracy: 0.63 (+/- 0.09)

```

Figure 2.10: Hasil Codingan No 10.

11. Penjelasan codingan ini mengartikan bahwa variabel `depth_acc` akan mengeksekusi `empty` dari importan library `numpy` yang dinamakan `buahpepaya` dengan 2 parameter yaitu 19,3 dan float. `i` didefinisikan dengan angka 0 kemudian untuk perhitungan jarak max depth diantara parameter 1 dan 20. Variabel `buahapel` mengartikan klasifikasi decision tree dengan 2 parameter. setelah itu, variabel `score` mendefinisikan variabel `depth_acc` dengan `i` dan 0, variabel kedua dari `depth_acc` dengan `i` dan 1 serta variabel ketiga dari `depth_acc` dengan `i` dan 2, maka pengeksekusian akhir bahwa variabel `i` akan ditambah dengan angka 1 untuk hasil akhirnya. Keluarannya akan berupa array dari perhitungan parameter dan variabel yang telah didefinisikan sebelumnya.

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.11.

12. Penjelasan codingan ini mendefinisikan pemanggilan dari library `matplotlib.pyplot` sebagai `buahanggur` sehingga nanti hasilnya akan berbentuk gambar grafik/gelombang. Untuk variabel `fig` dan `ax` akan mendefinisikan subplots dari `buahanggur`. Setelah itu ketentuan dari parameter `depth acc = 0`, `depth acc = 1` dan `depth acc 2`. Selanjutnya untuk menampilkan gelombang maka panggil variabel `buahanggur` dengan perintah `show`.

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.12.

```

In [12]: depth_acc = buahpepaya.empty((19,3), float)
...: i = 0
...: for max_depth in range(1, 20):
...:     buahapel = tree.DecisionTreeClassifier(criterion="entropy", max_depth=max_depth)
...:     scores = cross_val_score(buahapel, buahpir_att, buahpir_pass, cv=5)
...:     depth_acc[i,0] = max_depth
...:     depth_acc[i,1] = scores.mean()
...:     depth_acc[i,2] = scores.std() * 2
...:     i += 1
...:
...:
...:
...: depth_acc
Out[12]:
array([[ 1.      ,  0.63795172,  0.02257095],
       [ 2.      ,  0.68720762,  0.02295034],
       [ 3.      ,  0.69178704,  0.06621384],
       [ 4.      ,  0.69181089,  0.05105217],
       [ 5.      ,  0.67026014,  0.02969374],
       [ 6.      ,  0.66411731,  0.06858724],
       [ 7.      ,  0.68263885,  0.06498788],
       [ 8.      ,  0.67186961,  0.04285183],
       [ 9.      ,  0.65182173,  0.03602718],
       [10.     ,  0.65173861,  0.04048406],
       [11.     ,  0.65169145,  0.07924518],
       [12.     ,  0.63476783,  0.04265046],
       [13.     ,  0.62398685,  0.06238911],

```

Figure 2.11: Hasil Codingan No 11.

### 2.1.3 Penanganan Error

1. ScreeShootan Error pada codingan No 8 dapat dilihat pada gambar 2.13.
2. Codingan error dan jenis erornya : sebenarnya tidak terdapat error pada codingan ini namun saat pertama kali di run current cell codingan ini akan error dan tidak keluar outputannya dikarenakan library graphviz sebelumnya tidak ditemukan atau belum di install terlebih dahulu.

```

import graphviz
dot_data = tree.export_graphviz(buahapel, out_file=None, label="all", impurity=
                                feature_names=list(buahpir_train_att), class_names=
                                filled=True, rounded=True)

graph = graphviz.Source(dot_data)
graph

```

3. Solusi pemecahan masalah error tersebut yaitu dengan cara menginstall terlebih dahulu library graphviznya pada anaconda prompt atau command prompt anda

```
In [13]: import matplotlib.pyplot as buahanggur
...: fig, ax = buahanggur.subplots()
...: ax.errorbar(depth_acc[:,0], depth_acc[:,1], yerr=depth_acc[:,2])
...: buahanggur.show()
```

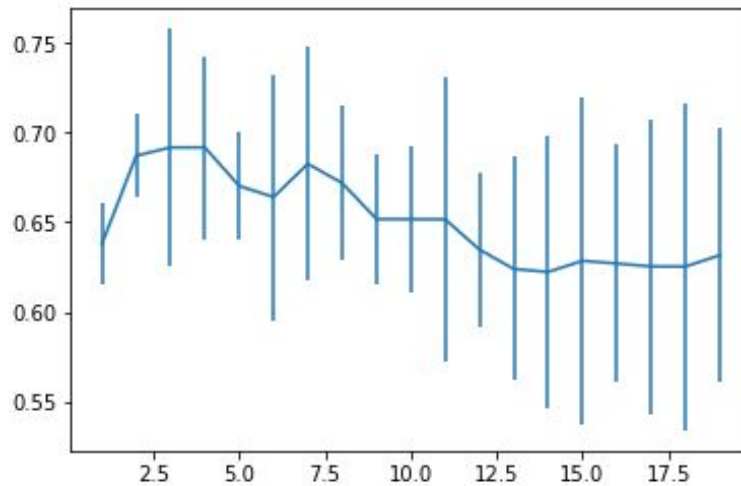


Figure 2.12: Hasil Codingan No 12.

dengan perintah conda install graphviz setelah itu run kembali codingan No 8 maka akan muncul outputan atau tampilan keluarannya.

Berikut gambar cara menginstall graphviz dapat dilihat pada gambar 2.14



```

In [25]: import graphviz
...: dot_data = tree.export_graphviz(buahpepaya, out_file=None, label="all",
...: impurity=False, proportion=True,
...: feature_names=list(buahpir_train_att),
...: class_names=["fail", "pass"],
...: filled=True, rounded=True)
...: graph = graphviz.Source(dot_data)
...: graph
Traceback (most recent call last):

  File "F:\anaconda\lib\site-packages\IPython\core\formatters.py", line 345, in __call__
    return method()

  File "F:\anaconda\lib\site-packages\graphviz\files.py", line 106, in _repr_svg_
    return self.pipe(format='svg').decode(self._encoding)

  File "F:\anaconda\lib\site-packages\graphviz\files.py", line 128, in pipe
    out = backend.pipe(self._engine, format, data, renderer, formatter)

  File "F:\anaconda\lib\site-packages\graphviz\backend.py", line 206, in pipe
    out, _ = run(cmd, input=data, capture_output=True, check=True, quiet=quiet)

  File "F:\anaconda\lib\site-packages\graphviz\backend.py", line 150, in run
    raise ExecutableNotFound(cmd)

ExecutableNotFound: failed to execute ['dot', '-Tsvg'], make sure the Graphviz executables are on your systems' PATH

```

Figure 2.13: Hasil Gambar Error No 6.

```

C:\Users\HUDA>conda install graphviz
Collecting package metadata: done
Solving environment: done

## Package Plan ##

  environment location: F:\anaconda

added / updated specs:
- graphviz

The following packages will be downloaded:

  package | build | size
  ----- | ----- | -----
  graphviz-2.38 | hfa6e2cd_3 | 37.7 MB
  ----- | ----- | -----
  Total: | | 37.7 MB

The following NEW packages will be INSTALLED:

  graphviz | pkgs/main/win-32::graphviz-2.38-hfa6e2cd_3

Proceed ([y]/n)?

Downloading and Extracting Packages
graphviz-2.38 | 37.7 MB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

```

Figure 2.14: Hasil Gambar Penanganan Error No 6.

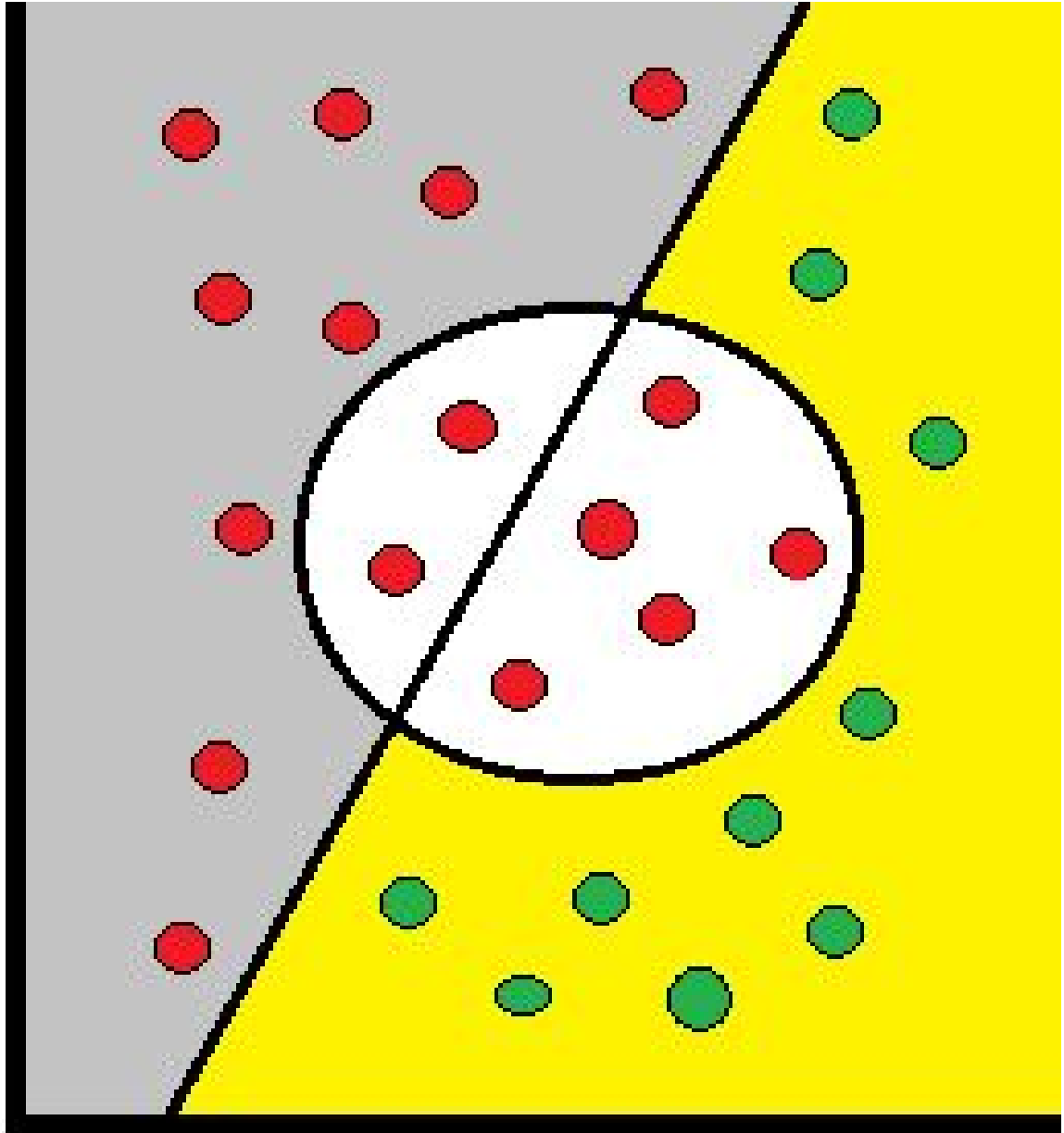


Figure 2.15: Binary Classification.

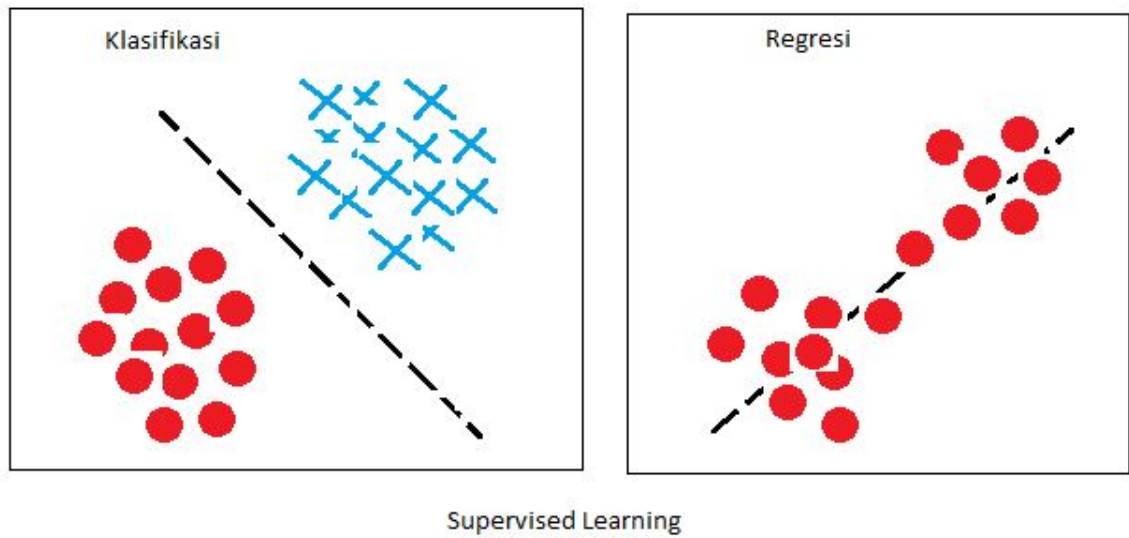


Figure 2.16: Supervised Learning.

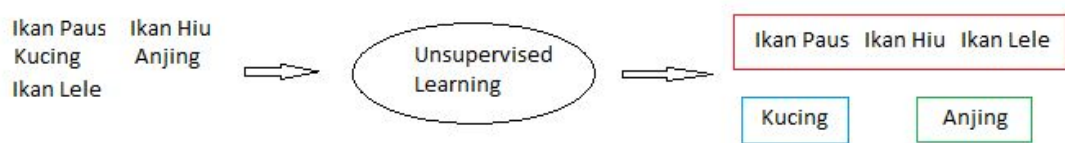


Figure 2.17: Unsupervised Learning.

## Clustering

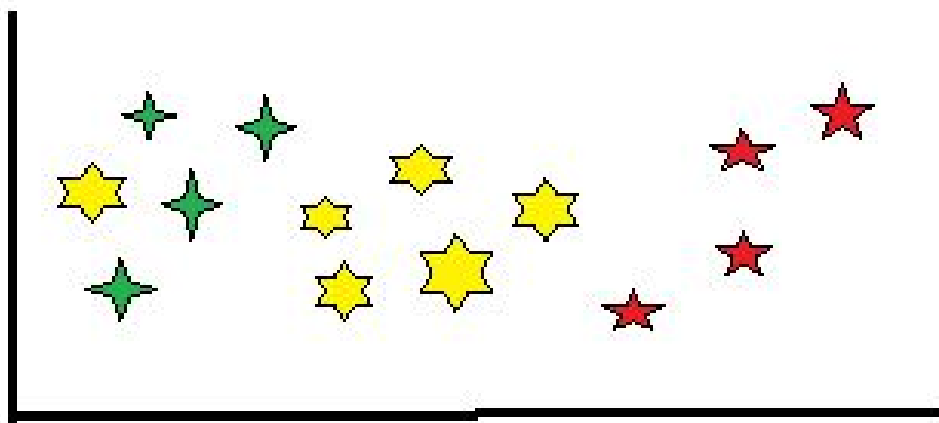


Figure 2.18: Clustering.

	Diprediksi Pena	Diprediksi Pensil
True Pena	10	5
True Pensil	7	8

Figure 2.19: Evaluasi dan Akurasi.

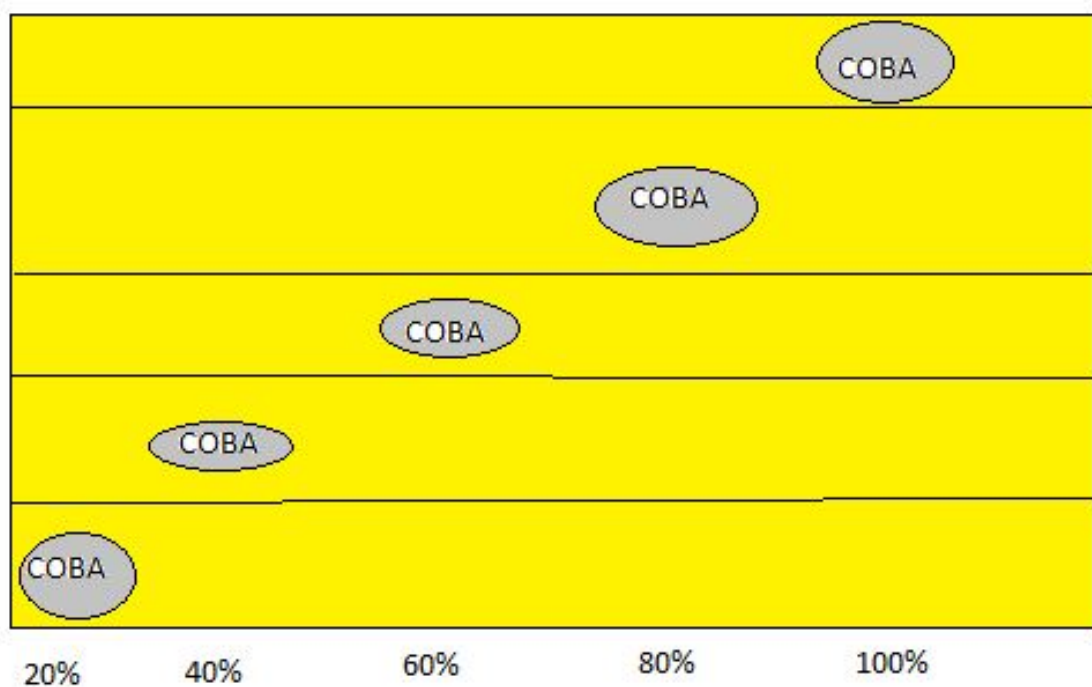


Figure 2.20: K-fold Cross Validation.



Figure 2.21: Decision Tree.

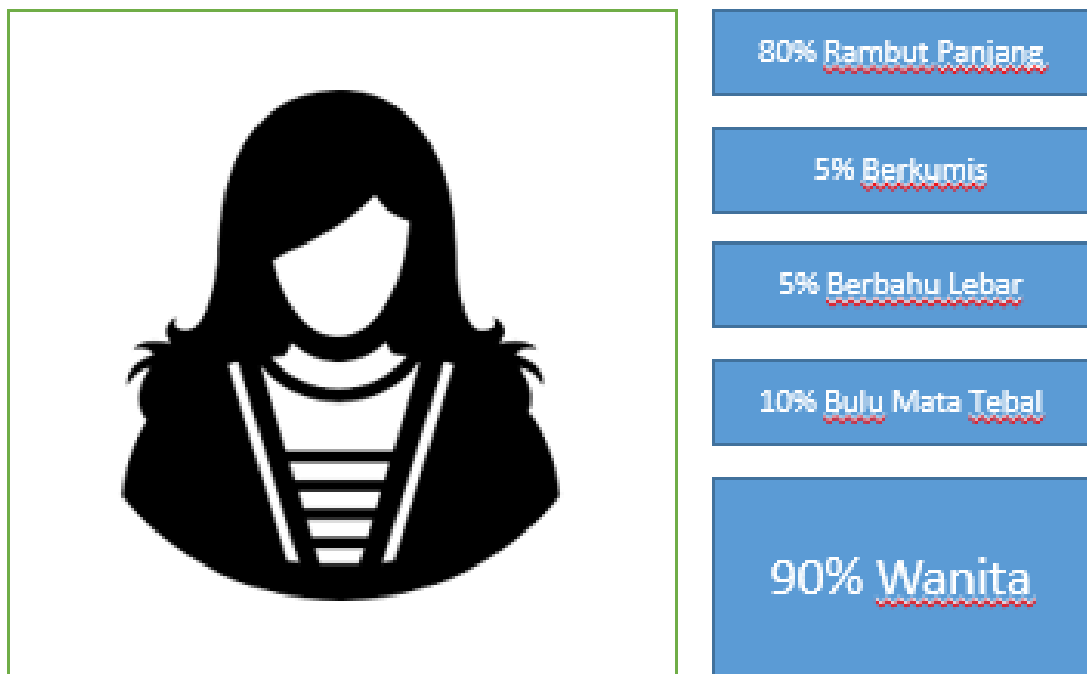


Figure 2.22: Gain.

# Chapter 3

## Methods

### 3.1 Ahmad Syafrizal Huda / 1164062

#### 3.1.1 Teori

1. Random forest ialah sekumpulan classifier yang terdiri dari banyak pohon keputusan dan mengerjakan klasifikasi berdasarkan keluaran dari hasil klasifikasi setiap pohon keputusan anggota. Klasifikasi random forest dikerjakan melalui penggabungan pohon (tree) dengan melakukan training pada sampel data yang dimiliki. Contoh ilustrasi gambar random forest pada gambar 3.1.
2. Cara membaca dataset kasus dan makna setiap file dan isi field masing-masing file

Gunakan librari pandas yang sudah di install sebelumnya pada python untuk dapat membaca dataset dengan format text file.

Selanjutnya buatlah variabel baru misalkan diberi nama dataset yang berisikan perintah untuk membaca file csv.

```
import pandas as data
dataset = data.read_csv("car.txt")
dataset.head()
```

Pada perintah diatas yaitu memanggil library pandas untuk membaca dataset, membuat variabel dataset yang berisikan data.read\_csv untuk membaca dataset. Pada contoh ini menggunakan txt tapi tetap bisa membaca datasetnya, karena pada saat dijalankan librari pandas secara otomatis akan mengubah data dalam bentuk text file ke format csv. Hasilnya seperti pada gambar 3.2.

Penjelasan dari isi field pada hasil gambar 3.2 yaitu: Atribut Index merupakan atribut otomatis untuk penomoran data yang sudah ada, Atribut Buying merupakan harga beli dari mobil tersebut. dengan value : v high/Sangat mahal,high/mahal,med/Cukup, low/Murah, Atribut Maint merupakan harga perawatan dari mobil tersebut, dengan value sama seperti pada atribut Buying, Atribut Doors merupakan jumlah pintu yang terdapat pada mobil, dengan value 2,3,4,5 more atau lebih dari 5, Atribut Persons merupakan kapasitas orang yang bisa masuk kedalam mobil, dengan value 2,4, more /lebih, Atribut Lug Boot merupakan ukuran bagasi boot mobil, dengan value small,med,big, Atribut Safety merupakan perkiraan keselamatan mobil, dengan value low,med,high, Yang terakhir yaitu Value, yang dimana merupakan merupakan Class nya atau disebut dengan targetnya menyatakan apakah mobil tersebut dapat diterima atau tidak dan apakah mobil tersebut bagus atau tidak, dengan value unacc, acc, good,v good.

3. Cross validation adalah teknik validasi model untuk menilai bagaimana hasil analisis statistik (model) akan digeneralisasi ke kumpulan data independen. Ini terutama digunakan dalam pengaturan di mana tujuannya adalah prediksi, dan orang ingin memperkirakan seberapa akurat model prediksi akan dilakukan dalam praktek.
4. Arti score 44% pada random forest yaitu merupakan outputannya untuk hutan acak, arti score 27% pada decision tree adalah presentasi hasil dari perhitungan dataset acak, dan arti score 29% dari SVM adalah hasil pendekatan jaringan saraf. Hasil tersebut didapat dari hasil validasi silang untuk memastikan bahwa membagi training test dengan cara yang berbeda. Jadi 44% untuk random forest, 27% untuk pohon keputusan, dan 29% untuk SVM. Itu merupakan presentase keakurasian prediksi yang dilakukan pada saat testing menggunakan label pada dataset yang digunakan. Score mendefinisikan aturan evaluasi model.
5. Perhitungan confusion Matriks dapat dilakukan sebagai berikut:

Import librari Pandas, Matplotlib, dan Numpy.

Buat variabel x\_aktu yang berisikan data aktual.

Buat variabel x\_diksi berisikan data yang akan dijadikan sebagai prediksi.

Buat variabel ak\_confusion yang berisikan crosstab untuk membangun tabel tabulasi silang yang dapat menunjukkan frekuensi kemunculan kelompok data tertentu.

Pada variabel `ak_confusion` definisikan lagi nama baris yaitu Aktual dan kolomnya Prediksi

Kemudian definisikan suatu fungsi yang diberi nama `plot_confusion_matrix` yang berisikan pendefinisian confusion matrix dan juga akan di plotting. untuk code perintah lengkapnya sebagai berikut :

```
import numpy as cm1
import matplotlib.pyplot as plt
import pandas as cm2
x_aktu = cm2.Series([3, 1, 3, 3, 1, 2, 2, 3, 3, 1, 2, 3], name='Aktual')
x_diksi = cm2.Series([1, 1, 3, 2, 1, 3, 2, 1, 3, 1, 3, 3], name='Prediksi')
ak_confusion = cm2.crosstab(x_aktu, x_diksi)
ak_confusion = cm2.crosstab(x_aktu, x_diksi, rownames=['Aktual'], colnames=['Prediksi'])
def plot_confusion_matrix(df_confusion, title='Confusion matrix', cmap=plt.cm.Blues):
    plt.matshow(ak_confusion, cmap=cmap) # imshow
    plt.title(title)
    plt.colorbar()
    tick_marks = cm1.arange(len(ak_confusion.columns))
    plt.xticks(tick_marks, ak_confusion.columns, rotation=45)
    plt.yticks(tick_marks, ak_confusion.index)
    plt.tight_layout()
    plt.ylabel(ak_confusion.index.name)
    plt.xlabel(ak_confusion.columns.name)
plot_confusion_matrix(ak_confusion)
plt.show()
```

Hasilnya akan seperti pada gambar 3.3 :

6. Voting pada random forest sebagaimana voting yaitu suara/hasil untuk setiap target yang diprediksi pada saat melakukan Random Forest. Pertimbangkan target prediksi dengan voting terbanyak/tertinggi sebagai prediksi akhir dari algoritma random forest. Contoh ilustrasi dapat dilihat pada gambar 3.4.



### 3.1.2 Praktek Program

```
1. import pandas as huda
   a = huda.Series(['Ahmad', 'Syaf', 'Rizal', 'Huda', 'Mubarrok'],
   index = [1, 2, 3, 4, 5])
   print (a)
```

Baris pertama pada codingan, yaitu import pandas as huda yang artinya kita akan mengimport librari pandas dari python dengan inisiasi huda.

Baris kedua pada codingan, yaitu Variabel a didefinisikan data data yang sudah dibuat seperti daftar nama dengan menggunakan huda.Series. Series adalah object satu dimensi yang serupa dengan kolom di dalam tabel.

Baris ketiga pada codingan, yaitu index digunakan untuk melabeli data dengan dimulai dari nomor 1...5, jika tidak label default akan dimulai dari 0,1,2...

Baris keempat pada codingan, yaitu digunakan untuk mencetak atau menampilkan data pada variabel a yang sudah dibuat sebelumnya.

Untuk hasilnya dapat dilihat pada gambar 3.5.

```
2. import numpy as huda
   print (huda.arange(50, 101))
```

Baris pertama pada codingan, yaitu import numpy as huda yang artinya kita akan mengimport librari numpy dari python dengan inisiasi huda.

Baris kedua pada codingan, yaitu digunakan untuk mencetak atau menampilkan data dengan menggunakan huda.arange untuk membuat array dengan bilangan sekuensial dari mulai nilai awal 50 sampai sebelum 101 dengan berurutan.

Untuk hasilnya dapat dilihat pada gambar 3.6.

```
3. import matplotlib.pyplot as huda
   huda.plot([0,1,3,5,7,8,10],[25, 35, 30, 45, 50, 45, 30])
   huda.show()
```

Baris pertama pada codingan, yaitu import matplotlib.pyplot as huda yang artinya kita akan mengimport librari matplotlib dengan class pyplot dari python dengan inisiasi huda.

Baris kedua pada codingan, yaitu digunakan untuk memasukkan data dengan insiai huda pada class plot dengan nilai x dan y yg sudah ditentukan.

Baris ketiga pada codingan, yaitu digunakan untuk mencetak atau menampilkan data dengan inisiasi `huda.show` nantinya keluarannya berupa grafik.

Untuk hasilnya dapat dilihat pada gambar 3.7.

#### 4. Menjalankan Klasifikasi Random Forest.

Pada gambar 3.8 berfungsi untuk membaca data yang berupa `image_attribute_labels` dengan format text file. Dengan mendefinisikan variabel `imgatt` yang berisikan value untuk membaca data, juga menggunakan code untuk skip data yang mengandung bad lines agar tidak terjadi error pada saat pembacaan file.

Pada gambar 3.9 yaitu mengembalikan baris `n` teratas (5 secara default) dari dataframe `imgatt`.

Pada gambar 3.10 yaitu menampilkan beberapa baris dan kolom dari dataframe `imgatt`.

Pada gambar 3.11 yaitu variabel `imgatt2` menggunakan function `pivot` untuk mengubah kolom jadi baris, dan baris jadi kolom dari dataframe sebelumnya.

Pada gambar 3.12 yaitu `imgatt2 head` berfungsi untuk mengembalikan nilai teratas dari dataframe `imgatt2`.

Pada gambar 3.13 yaitu menampilkan beberapa baris dan kolom dari dataframe `imgatt2`.

Pada gambar 3.14 yaitu melakukan `pivot` yang mana `imgid` menjadi index yang artinya unik.

Pada gambar 3.15 yaitu memuat jawabannya yang berisi apakah burung itu termasuk dalam spesies yang mana. Dua kolomnya terdiri dari `imgid` dan label.

Pada gambar 3.16 yaitu menunjukkan 11788 baris dan 1 kolom. Dimana kolom itu adalah jenis spesies burungnya.

Pada gambar 3.17 yaitu `join` antara `imgatt2` dengan `imglabels` dikarenakan isinya sama. Sehingga kita bisa mendapatkan data ciri dan data jawabannya/labelnya sehingga bisa dikategorikan supervised learning.

Pada gambar 3.18 yaitu menghilangkan label yang didepan, dan menggunakan label yang paling belakang yang baru di `join`.

Pada gambar 3.19 yaitu mengecek 5 data teratas dari `df att`.

Pada gambar 3.20 yaitu mengecek 5 data teratas dari `df label`.

Pada gambar 3.21 yaitu 8000 row pertama sebagai data training sisanya sebagai data testing dengan membagi menjadi dua bagian.

Pada gambar 3.22 yaitu memanggil kelas RandomForestClassifier. max features diartikan sebagai berapa banyak kolom pada setiap tree dengan isi maksimum 50.

Pada gambar 3.23 yaitu melakukan fit untuk membangun random forest yang sudah ditentukan dengan maksimum fitur sebanyak 50 untuk perpohonnya.

Pada gambar 3.24 yaitu menampilkan hasil prediksi dari random forest sebelumnya.

Pada gambar 3.25 yaitu menampilkan besaran akurasi dari prediksi diatas atau score perolehan dari klasifikasi.

#### 5. Menjalankan Confusion Matrix.

Pada gambar 3.26 yaitu menyatukan Random Forest ke dalam Confusion Matrix

Pada gambar 3.27 yaitu menampilkan hasil dari gambar sebelumnya dengan array pada perintah cm.

Pada gambar 3.28 yaitu Plotting Confusion Matrix dengan librari Matplotlib.

Pada gambar 3.29 yaitu Set plot sumbunya sesuai dengan nama datanya dan membaca file classes.txt.

Pada gambar 3.30 yaitu Plot hasil perubahan label yang sudah dilakukan.

#### 6. Menjalankan Klasifikasi SVM dan Decision Tree.

Pada gambar 3.31 yaitu klasifikasi dengan decision tree dengan dataset yang sama dan akan muncul akurasi prediksinya.

Pada gambar 3.32 yaitu klasifikasi dengan SVM dengan dataset yang sama dan akan muncul akurasi prediksinya.

#### 7. Menjalankan Cross Validation.

Pada gambar 3.33 yaitu Cross Validation untuk Random Forest.

Pada gambar 3.34 yaitu Cross Validation untuk Decision Tree.

Pada gambar 3.35 yaitu Cross Validation untuk SVM.

## 8. Menjalankan Pengamatan Komponen Informasi.

Pada gambar 3.36 yaitu untuk mengetahui berapa banyak tree yang dibuat, berapa banyak atribut yang dipakai dan informasi lainnya.

Pada gambar 3.37 yaitu hasil dari plotting komponen informasi agar dapat dibaca.

### 3.1.3 Penanganan Error

#### 1. Screenshootan error ada pada gambar 3.38.

#### 2. from sklearn.model\_selection import cross\_val\_score

```
scores = cross_val_score(clf, df_train_att, df_train_label, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
scorestree = cross_val_score(clftree, df_train_att, df_train_label, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(), scorestree.std() * 2))
```

```
scoressvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(), scoressvm.std() * 2))
```

```
max_features_opts = range(1, 10, 1)
```

```
n_estimators_opts = range(2, 40, 4)
```

```
rf_params = np.empty((len(max_features_opts)*len(n_estimators_opts),4), float)
i = 0
```

```
for max_features in max_features_opts:
```

```
    for n_estimators in n_estimators_opts:
```

```
        clf = RandomForestClassifier(max_features=max_features, n_estimators=
```

```
        scores = cross_val_score(clf, df_train_att, df_train_label, cv=5)
```

```
        rf_params[i,0] = max_features
```

```
        rf_params[i,1] = n_estimators
```

```
        rf_params[i,2] = scores.mean()
```

```
        rf_params[i,3] = scores.std() * 2
```

```
        i += 1
```

```
print("Max features: %d, num estimators: %d, accuracy: %0.2f (+/- %0.2f)" %
```

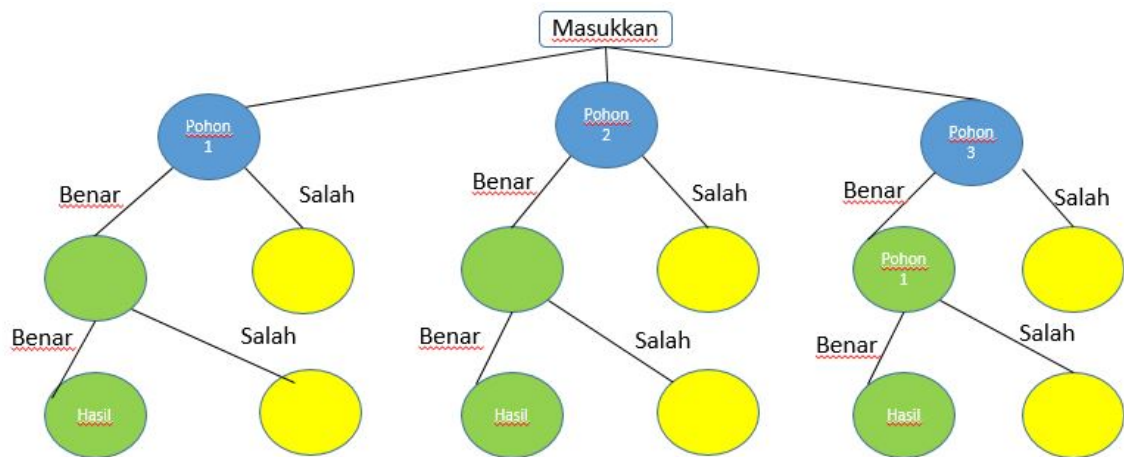


Figure 3.1: Random Forest

3. Solusi pemecahan masalah tersebut yaitu dengan cara mengubah pada codingan dibawah ini yang awalnya datanya 8000 menjadi 1000 semua dan max\_featurenya awalnya 50 menjadi 25.

```

df_train_att = df_att[:1000]
df_train_label = df_label[:1000]
df_test_att = df_att[1000:]
df_test_label = df_label[1000:]

df_train_label = df_train_label['label']
df_test_label = df_test_label['label']

from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(max_features=25, random_state=0, n_estimators=100)

```

Jika sudah dilakukan maka langkah selanjutnya yaitu kita tinggal merunning ulang dari awal hingga data tersebut berhasil diajalankan

dataset - DataFrame							
Index	buying	maint	doors	persons	lug_boot	safety	value
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc
5	vhigh	vhigh	2	2	med	high	unacc
6	vhigh	vhigh	2	2	big	low	unacc
7	vhigh	vhigh	2	2	big	med	unacc
8	vhigh	vhigh	2	2	big	high	unacc
9	vhigh	vhigh	2	4	small	low	unacc
10	vhigh	vhigh	2	4	small	med	unacc
11	vhigh	vhigh	2	4	small	high	unacc
12	vhigh	vhigh	2	4	med	low	unacc
13	vhigh	vhigh	2	4	med	med	unacc
14	vhigh	vhigh	2	4	med	high	unacc
15	vhigh	vhigh	2	4	big	low	unacc
16	vhigh	vhigh	2	4	big	med	unacc
17	vhigh	vhigh	2	4	big	high	unacc
18	vhigh	vhigh	2	more	small	low	unacc
19	vhigh	vhigh	2	more	small	med	unacc
20	vhigh	vhigh	2	more	small	high	unacc
21	vhigh	vhigh	2	more	med	low	unacc

Figure 3.2: Hasil Dataset

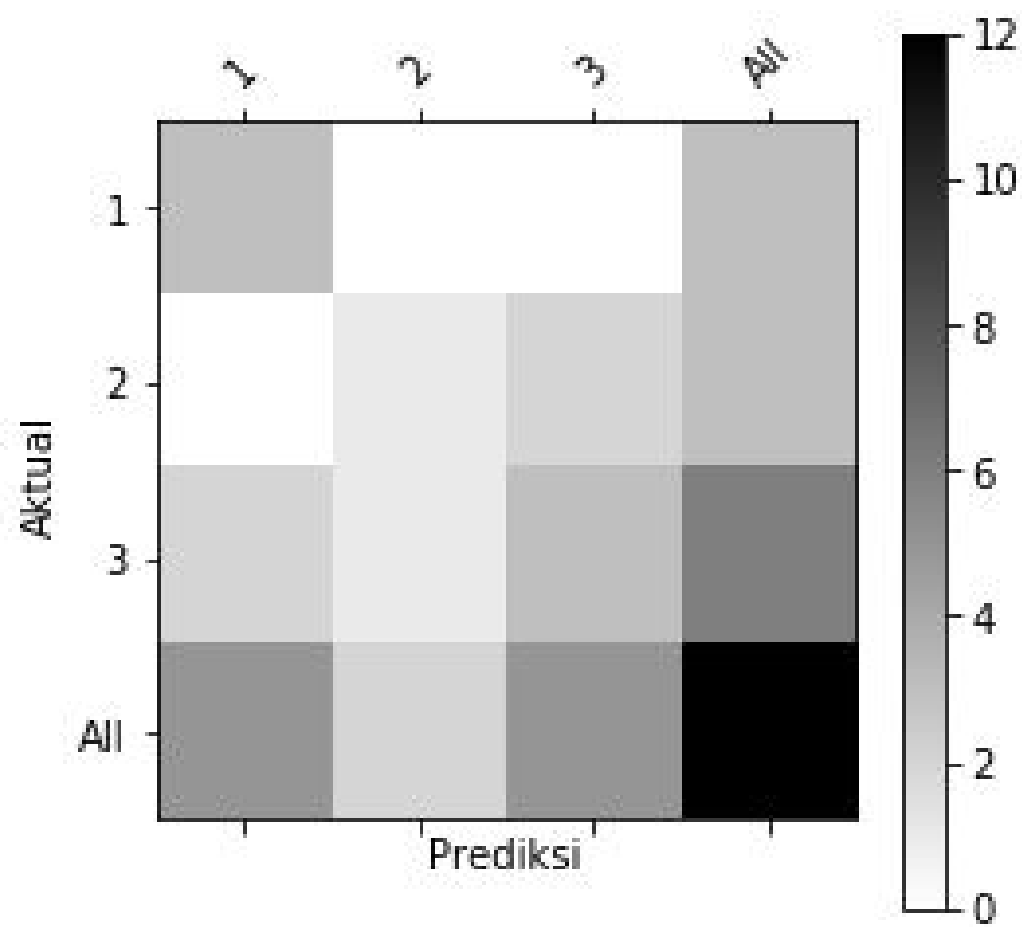


Figure 3.3: Confusion Matrix

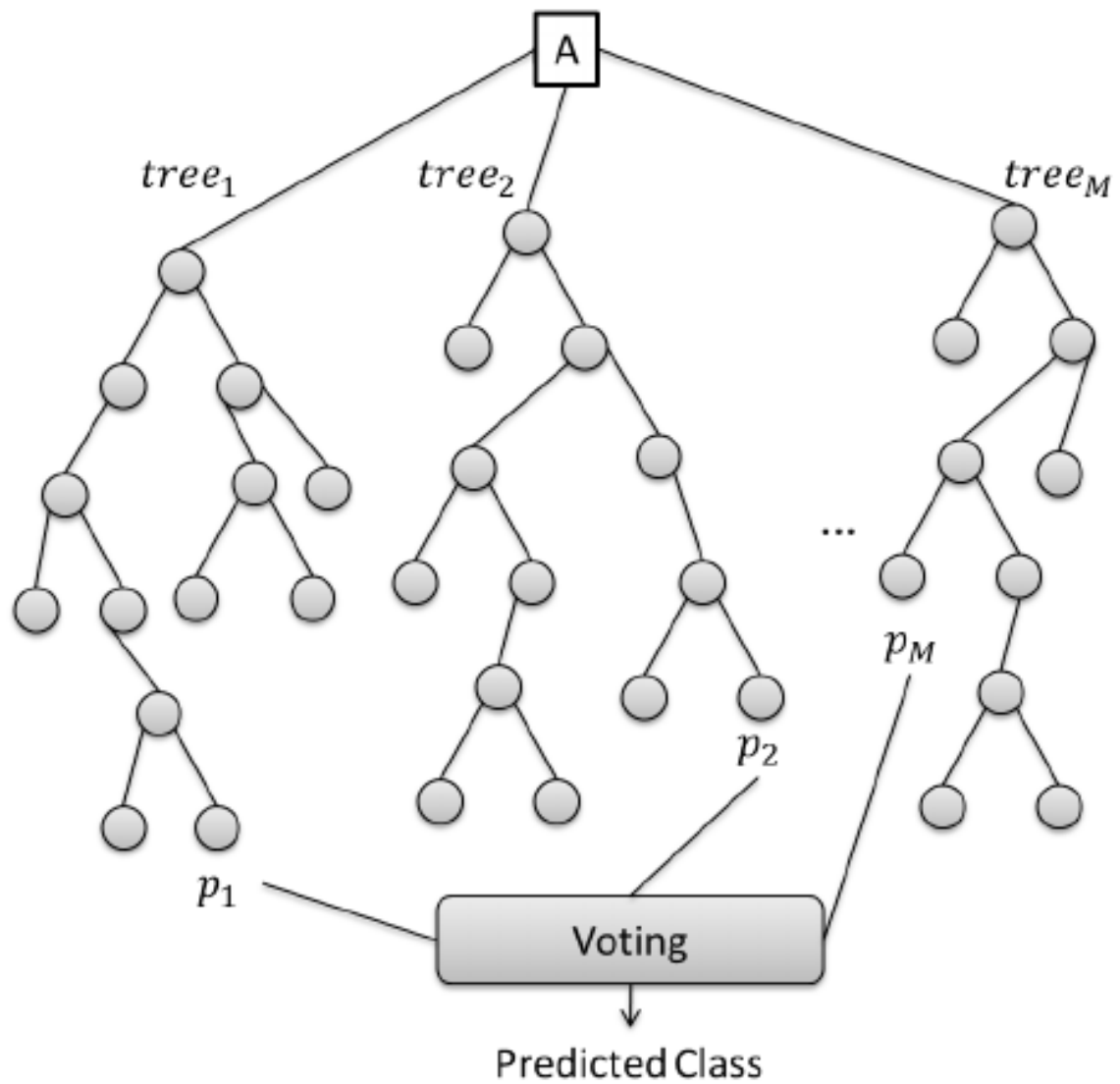


Figure 3.4: Voting

```
In [60]: import pandas as huda
...: a = huda.Series(['Ahmad', 'Syaf', 'Rizal', 'Huda', 'Mubarrok'],
...: index = [1, 2, 3, 4, 5])
...: print (a)
1      Ahmad
2      Syaf
3      Rizal
4      Huda
5  Mubarrok
dtype: object
```

Figure 3.5: Aplikasi Menggunakan Pandas



```
In [61]: import numpy as huda
...: print (huda.arange(50, 101))
```

```
[ 50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67
  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85
  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100]
```

Figure 3.6: Aplikasi Menggunakan Numpy

```
In [62]: import matplotlib.pyplot as huda
...: huda.plot([0,1,3,5,7,8,10],[25, 35, 30, 45, 50, 45, 30])
...: huda.show()
```

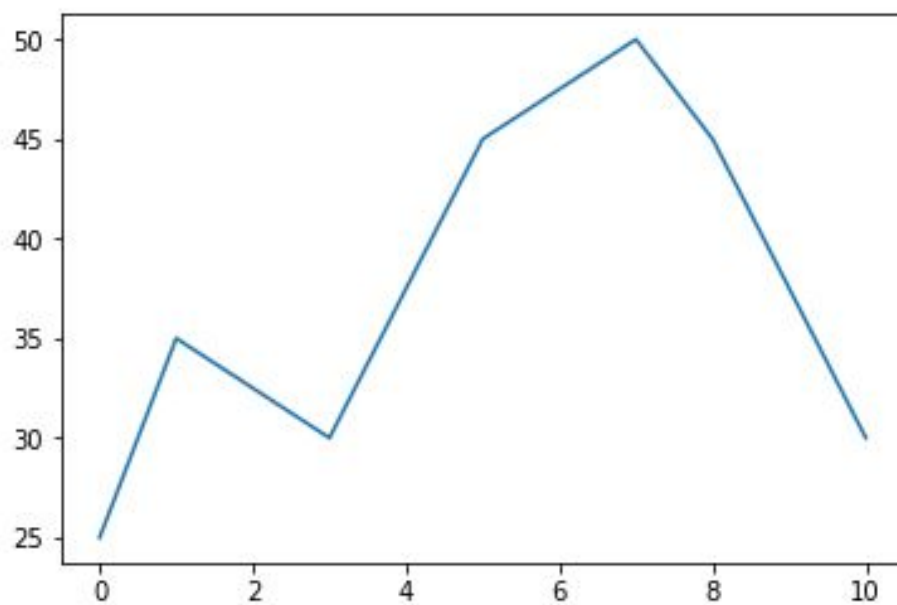


Figure 3.7: Aplikasi Menggunakan Matplotlib

Name	Type	Size	Value
imgatt	DataFrame	(3677856, 3)	Column names: imgid, attid, present

Figure 3.8: Hasil 1 Random Forest

```

In [2]: imgatt.head()
Out[2]:
   imgid  attid  present
0       1      1        0
1       1      2        0
2       1      3        0
3       1      4        0
4       1      5        1

```

Figure 3.9: Hasil 2 Random Forest

```

In [3]: imgatt.shape
Out[3]: (3677856, 3)

```

Figure 3.10: Hasil 3 Random Forest

Name	Type	Size	Value
imgatt	DataFrame	(3677856, 3)	Column names: imgid, attid, present
imgatt2	DataFrame	(11788, 312)	Column names: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1 ...

Figure 3.11: Hasil 4 Random Forest

```

In [5]: imgatt2.head()
Out[5]:
attid  1    2    3    4    5    6    7    ...    306  307  308  309  310  311  312
imgid
1      0    0    0    0    1    0    0    ...    0    0    1    0    0    0    0
2      0    0    0    0    0    0    0    ...    0    0    0    0    0    0    0
3      0    0    0    0    1    0    0    ...    0    0    1    0    0    1    0
4      0    0    0    0    1    0    0    ...    1    0    0    1    0    0    0
5      0    0    0    0    1    0    0    ...    0    0    0    0    0    0    0

[5 rows x 312 columns]

```

Figure 3.12: Hasil 5 Random Forest

```

In [6]: imgatt2.shape
Out[6]: (11788, 312)

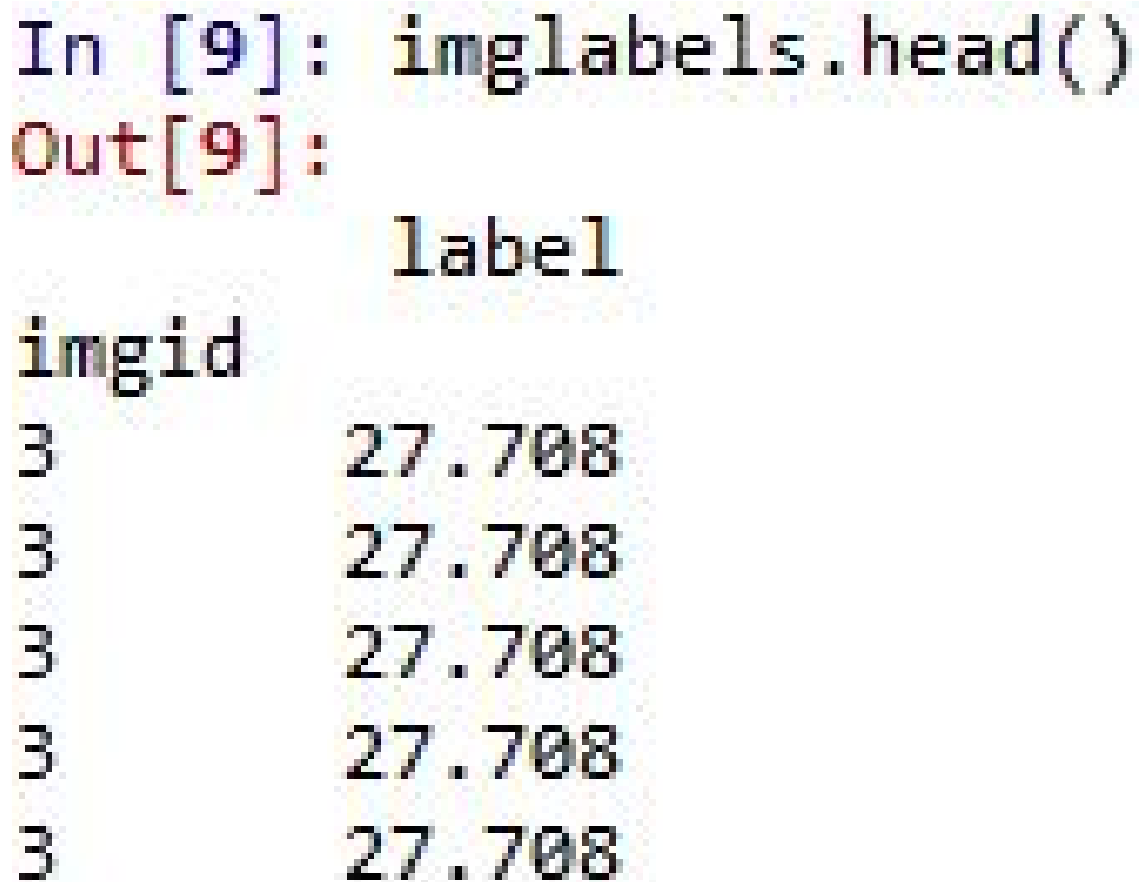
```

Figure 3.13: Hasil 6 Random Forest

Name	Type	Size	Value
imgatt	DataFrame	(3677856, 3)	Column names: imgid, attid, present
imgatt2	DataFrame	(11788, 312)	Column names: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1 ...
imglabels	DataFrame	(3677856, 1)	Column names: label

Figure 3.14: Hasil 7 Random Forest

```

In [9]: imglabels.head()
Out[9]:


| imgid | label  |
|-------|--------|
| 3     | 27.708 |
| 3     | 27.708 |
| 3     | 27.708 |
| 3     | 27.708 |
| 3     | 27.708 |


```

Figure 3.15: Hasil 8 Random Forest

```

In [10]: imglabels.shape
Out[10]: (3677856, 1)

```

Figure 3.16: Hasil 9 Random Forest

Name	Type	Size	Value
df	DataFrame	(11788, 313)	Column names: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1 ...
imgatt	DataFrame	(3677856, 3)	Column names: imgid, attid, present
imgatt2	DataFrame	(11788, 312)	Column names: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1 ...
imglabels	DataFrame	(11788, 1)	Column names: label

Figure 3.17: Hasil 10 Random Forest

Name	Type	Size	Value
df	DataFrame	(11788, 313)	Column names: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1 ...
df_att	DataFrame	(11788, 312)	Column names: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1 ...
df_label	DataFrame	(11788, 1)	Column names: label
imgatt	DataFrame	(3677856, 3)	Column names: imgid, attid, present
imgatt2	DataFrame	(11788, 312)	Column names: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1 ...

Figure 3.18: Hasil 11 Random Forest

```
In [13]: df_att.head()
Out[13]:
```

	1	2	3	4	5	6	7	...	306	307	308	309	310	311	312
imgid								...							
11436	0	0	0	0	0	0	1	...	0	0	0	1	0	0	0
9629	0	0	0	0	0	0	1	...	0	0	0	1	0	0	0
1226	0	0	0	0	0	0	1	...	0	0	1	0	0	1	0
3128	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1
5720	0	0	0	0	0	0	1	...	0	0	0	0	0	1	0

[5 rows x 312 columns]

Figure 3.19: Hasil 11 Random Forest

```

In [14]: df_label.head()
Out[14]:
      label
imgid
11436    195
9629     164
1226      22
3128      54
5720      98

```

Figure 3.20: Hasil 12 Random Forest

Name	Type	Size	Value
df_test_att	DataFrame	(3788, 312)	Column names: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1 ...
df_test_label	Series	(3788,)	Series object of pandas.core.series module
df_train_att	DataFrame	(8000, 312)	Column names: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1 ...
df_train_label	Series	(8000,)	Series object of pandas.core.series module

Figure 3.21: Hasil 13 Random Forest

```

In [15]: from sklearn.ensemble import RandomForestClassifier
...: clf = RandomForestClassifier(max_features=50, random_state=0,
n_estimators=100)

```

Figure 3.22: Hasil 14 Random Forest

```
In [16]: clf.fit(df_train_att, df_train_label)
Out[16]:
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                        max_depth=None, max_features=50, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
                        oob_score=False, random_state=0, verbose=0, warm_start=False)
```

Figure 3.23: Hasil 15 Random Forest

```
In [17]: print(clf.predict(df_train_att.head()))
[199  87 132 119 150]
```

Figure 3.24: Hasil 16 Random Forest

```
In [18]: clf.score(df_test_att, df_test_label)
Out[18]: 0.4429778247096093
```

Figure 3.25: Hasil 17 Random Forest

pred_labels	int64	(3788,)	[ 22  44  83 ... 143 170  98]
-------------	-------	---------	-------------------------------

Figure 3.26: Hasil 1 Confusion Matrix

```
In [20]: cm
Out[20]:
array([[ 7,  0,  0, ...,  0,  0,  0],
       [ 0, 12,  0, ...,  0,  0,  0],
       [ 2,  0, 10, ...,  0,  0,  0],
       ...,
       [ 0,  0,  0, ...,  3,  0,  0],
       [ 0,  0,  0, ...,  0,  7,  0],
       [ 0,  0,  0, ...,  0,  0, 11]], dtype=int64)
```

Figure 3.27: Hasil 2 Confusion Matrix



```

In [22]: import matplotlib.pyplot as plt
...: import itertools
...: def plot_confusion_matrix(cm, classes,
...:                           normalize=False,
...:                           title='Confusion matrix',
...:                           cmap=plt.cm.Blues):
...:     """
...:     This function prints and plots the confusion matrix.
...:     Normalization can be applied by setting `normalize=True`.
...:     """
...:     if normalize:
...:         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
...:         print("Normalized confusion matrix")
...:     else:
...:         print('Confusion matrix, without normalization')
...:
...:     print(cm)
...:
...:     plt.imshow(cm, interpolation='nearest', cmap=cmap)
...:     plt.title(title)
...:     #plt.colorbar()
...:     tick_marks = np.arange(len(classes))
...:     plt.xticks(tick_marks, classes, rotation=90)
...:     plt.yticks(tick_marks, classes)
...:
...:     fmt = '.2f' if normalize else 'd'
...:     thresh = cm.max() / 2.

```

Figure 3.28: Hasil 3 Confusion Matrix



```

---
174             175.Pine_Warbler
175             176.Prairie_Warbler
176             177.Prothonotary_Warbler
177             178.Swainson_Warbler
178             179.Tennessee_Warbler
179             180.Wilson_Warbler
180             181.Worm_eating_Warbler
181             182.Yellow_Warbler
182             183.Northern_Waterthrush
183             184.Louisiana_Waterthrush
184             185.Bohemian_Waxwing
185             186.Cedar_Waxwing
186     187.American_Three_toed_Woodpecker
187             188.Pileated_Woodpecker
188             189.Red_bellied_Woodpecker
189             190.Red_cockaded_Woodpecker
190             191.Red_headed_Woodpecker
191             192.Downy_Woodpecker
192             193.Bewick_Wren
193             194.Cactus_Wren
194             195.Carolina_Wren
195             196.House_Wren
196             197.Marsh_Wren
197             198.Rock_Wren
198             199.Winter_Wren
199             200.Common_Yellowthroat
Name: birdname, Length: 200, dtype: object

```

Figure 3.29: Hasil 4 Confusion Matrix

---

```

[[0.41 0.    0.    ... 0.    0.    0.   ]
 [0.    0.48 0.    ... 0.    0.    0.   ]
 [0.11 0.    0.53 ... 0.    0.    0.   ]
 ...
 [0.    0.    0.    ... 0.2   0.    0.   ]
 [0.    0.    0.    ... 0.    0.35 0.   ]
 [0.    0.    0.    ... 0.    0.    0.79]]

```

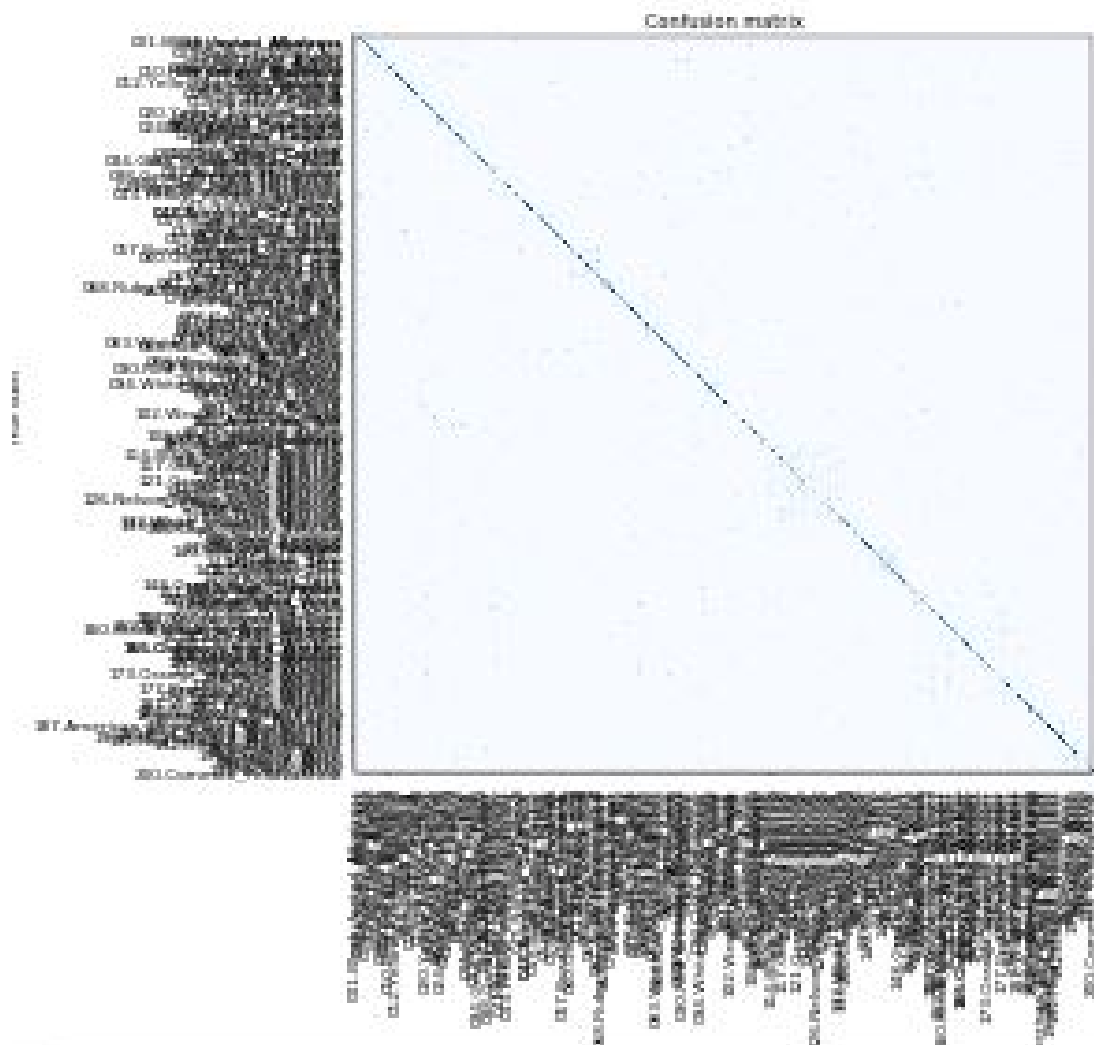


Figure 3.30: Hasil 5 Confusion Matrix

```

In [30]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(df_train_att, df_train_label)
...: clftree.score(df_test_att, df_test_label)
Out[30]: 0.28299894403379094

```

Figure 3.31: Hasil 1 Klasifikasi SVM dan Decision Tree

```

In [26]: from sklearn import svm
...: clfsvm = svm.SVC()
...: clfsvm.fit(df_train_att, df_train_label)
...: clfsvm.score(df_test_att, df_test_label)
F:\anaconda\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The default
value of gamma will change from 'auto' to 'scale' in version 0.22 to account better
for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this
warning.
  "avoid this warning.", FutureWarning)
Out[26]: 0.2808870116156283

```

Figure 3.32: Hasil 2 Klasifikasi SVM dan Decision Tree

```

In [26]: from sklearn.model_selection import cross_val_score
...: scores = cross_val_score(clf, df_train_att, df_train_label, cv=5)
...: # show average score and +/- two standard deviations away (covering 95% of
scores)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
F:\anaconda\lib\site-packages\sklearn\model_selection\_split.py:652: Warning: The
least populated class in y has only 1 members, which is too few. The minimum number
of members in any class cannot be less than n_splits=5.
  % (min_groups, self.n_splits)), Warning)
Accuracy: 0.30 (+/- 0.05)

```

Figure 3.33: Hasil 1 Cross Validation

```

In [27]: scorestree = cross_val_score(clftree, df_train_att, df_train_label, cv=5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(), scorestree.std()
* 2))
F:\anaconda\lib\site-packages\sklearn\model_selection\_split.py:652: Warning: The
least populated class in y has only 1 members, which is too few. The minimum number
of members in any class cannot be less than n_splits=5.
  % (min_groups, self.n_splits)), Warning)
Accuracy: 0.15 (+/- 0.04)

```

Figure 3.34: Hasil 2 Cross Validation

```

In [28]: scoressvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv=5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(), scoressvm.std() * 2))
F:\anaconda\lib\site-packages\sklearn\model_selection\_split.py:652: Warning: The least populated class in y has only 1 members, which is too few. The minimum number
of members in any class cannot be less than n_splits=5.
  % (min_groups, self.n_splits)), Warning)
F:\anaconda\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version 0.22 to account better
for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
F:\anaconda\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version 0.22 to account better
for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
F:\anaconda\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version 0.22 to account better
for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
F:\anaconda\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version 0.22 to account better
for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
F:\anaconda\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version 0.22 to account better
for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
Accuracy: 0.01 (+/- 0.00)

```

Figure 3.35: Hasil 3 Cross Validation

```

In [29]: max_features_opts = range(1, 10, 1)
...: n_estimators_opts = range(2, 40, 4)
...: rf_params = np.empty((len(max_features_opts)*len(n_estimators_opts),4), float)
...: i = 0
...: for max_features in max_features_opts:
...:     for n_estimators in n_estimators_opts:
...:         clf = RandomForestClassifier(max_features=max_features, n_estimators=n_estimators)
...:         scores = cross_val_score(clf, df_train_att, df_train_label, cv=5)
...:         rf_params[i,0] = max_features
...:         rf_params[i,1] = n_estimators
...:         rf_params[i,2] = scores.mean()
...:         rf_params[i,3] = scores.std() * 2
...:         i += 1
...:     print("Max features: %d, num estimators: %d, accuracy: %0.2f (+/- %0.2f)" %
...:           (max_features, n_estimators, scores.mean(), scores.std() *
2))
F:\anaconda\lib\site-packages\sklearn\model_selection\_split.py:652: Warning: The least populated class in y has only 1 members, which is too few. The minimum number
of members in any class cannot be less than n_splits=5.
  % (min_groups, self.n_splits)), Warning)
F:\anaconda\lib\site-packages\sklearn\model_selection\_split.py:652: Warning: The least populated class in y has only 1 members, which is too few. The minimum number
of members in any class cannot be less than n_splits=5.
  % (min_groups, self.n_splits)), Warning)
Max features: 1, num estimators: 2, accuracy: 0.06 (+/- 0.04)
Max features: 1, num estimators: 6, accuracy: 0.08 (+/- 0.03)
F:\anaconda\lib\site-packages\sklearn\model_selection\_split.py:652: Warning: The least populated class in y has only 1 members, which is too few. The minimum number
of members in any class cannot be less than n_splits=5.
  % (min_groups, self.n_splits)), Warning)
Max features: 1, num estimators: 10, accuracy: 0.12 (+/- 0.03)
F:\anaconda\lib\site-packages\sklearn\model_selection\_split.py:652: Warning: The least populated class in y has only 1 members, which is too few. The minimum number
of members in any class cannot be less than n_splits=5.
  % (min_groups, self.n_splits)), Warning)

```

Figure 3.36: Hasil 1 Pengamatan Komponen Informasi

```

....: ax = fig.gca(projection='3d')
....: x = rf_params[:,0]
....: y = rf_params[:,1]
....: z = rf_params[:,2]
....: ax.scatter(x, y, z)
....: ax.set_zlim(0.02, 0.3)
....: ax.set_xlabel('Max features')
....: ax.set_ylabel('Num estimators')
....: ax.set_zlabel('Avg accuracy')
....: plt.show()

```

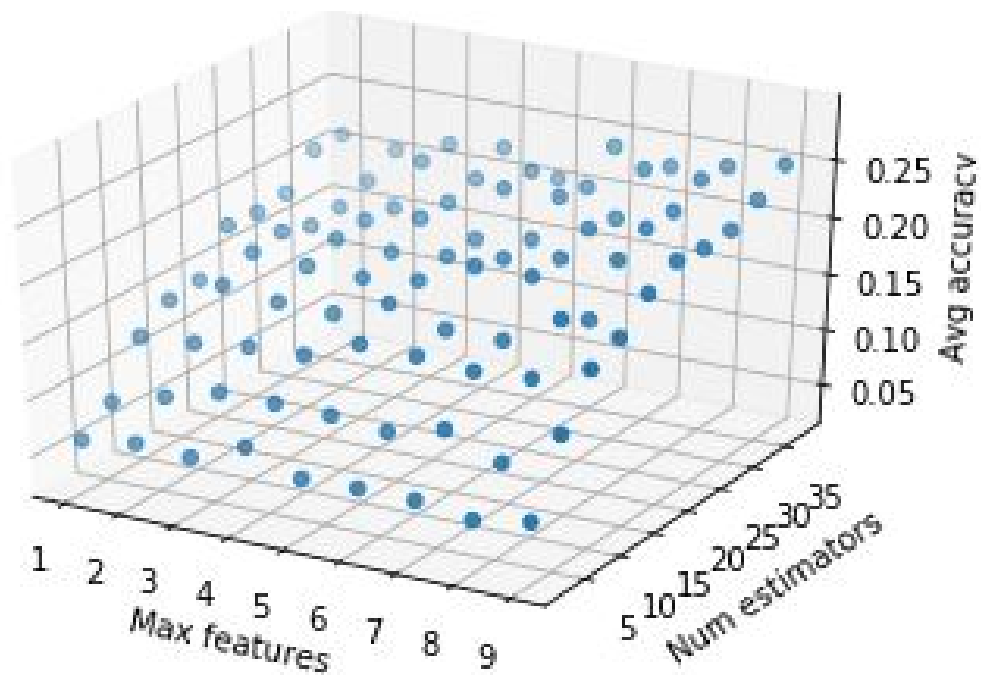


Figure 3.37: Hasil 2 Pengamatan Komponen Informasi

**MemoryError**

Figure 3.38: Error

# Chapter 4

## Experiment and Result

brief of experiment and result.

### 4.1 Experiment

Please tell how the experiment conducted from method.

### 4.2 Result

Please provide the result of experiment

### 4.3 Ahmad Syafrizal Huda/1164062

#### 4.3.1 Teori

1. Klasifikasi teks adalah proses pemberian kategori ke dalam teks/dokumen sesuai dengan tipikal dalam supervised machine learning (ML) yang bisa berupa buku perpustakaan, halaman web, artikel media, galeri, dan lain sebagainya. Tujuannya untuk memberikan label pada setiap teks/dokumen.  
Contoh ilustrasi gambar dapat dilihat pada gambar 4.1
2. Mengapa klasifikasi bunga tidak dapat menggunakan machine learning? itu dikarenakan memiliki masalah masukan(input) yang sama tetapi keluarannya (output) yang berbeda, biasanya output yang berbeda ini disebut dengan istilah 'noise'.

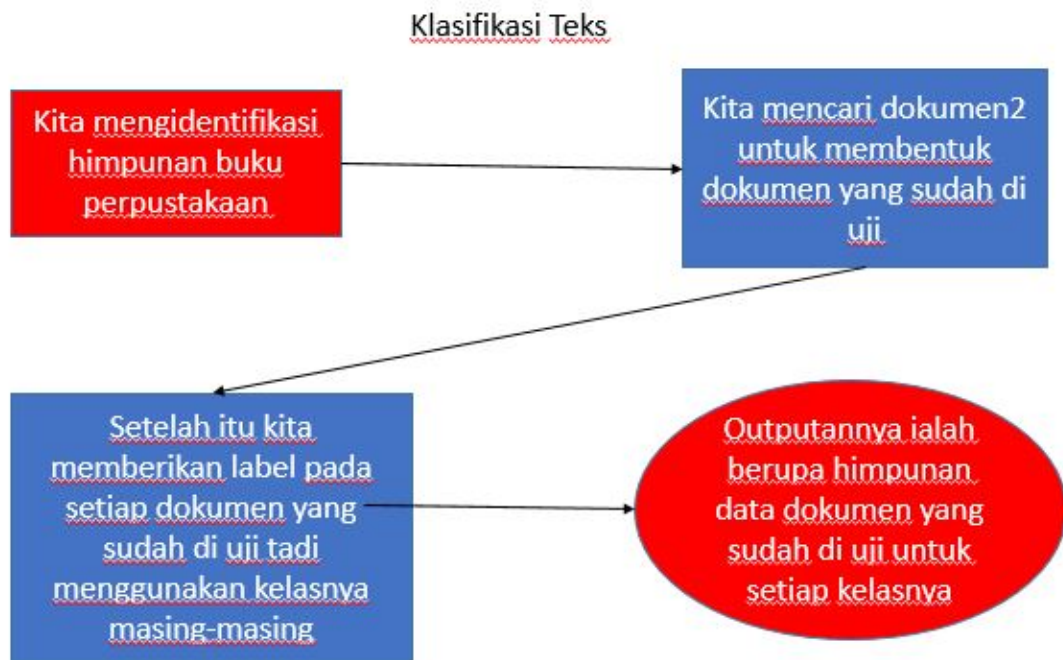


Figure 4.1: Klasifikasi Teks

Noise berarti output yang disimpan bukan seperti seharusnya (keluaran yang tidak diinginkan).

Contoh ilustrasi gambar dapat dilihat pada gambar 4.2

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa

Figure 4.2: Klasifikasi Bunga

3. Teknik pembelajaran mesin pada teks biasanya menggunakan teknik bag-of-words pada klasifikasi berbasis text dan kata untuk mengklasifikasikan komentar yang ada diinternet sebagai spam atau bukan. Misalkan pada kolom komentar di youtube dapat di cek seberapa sering suatu kata muncul dalam kalimat. Setiap kata dapat dijadikan baris dan kolom yang



merupakan kategori kata tersebut, apakah masuk kedalam spam atau tidak. dan contoh lainnya yaitu pada Caption. dimana akan muncul subtitle secara otomatis dari youtube menggunakan sensor suara yang disesuaikan dengan kata yang telah ditentukan.

Contoh ilustrasi gambar dapat dilihat pada gambar 4.3

348	z12he50arvrkivl5u04cctawgxzkjfsjcc4	2015-06-05T14:14:48	hi guys please my android photo editor downloa...	1
349	z13vhvu54u3ewpp5h04ccb4zuoardrmjlyk0k	2015-06-05T18:05:16	The first billion viewed this because they tho...	0

Figure 4.3: Comment Youtube

4. Vektorisasi data merupakan pembagian dan pemecahan data yang kemudian nantinya data tersebut akan menjadi beberapa data

Contoh misalkan dari sebuah paragraph nantinya kan di pecah menjadi beberapa kalimat dari kalimat tersebut dibagi lagi menjadi beberapa kata.

5. Bag-of-words adalah cara untuk merepresentasikan data teks saat memodelkan teks yang menggambarkan kemunculan

kata-kata dalam dokumen dengan algoritma pembelajaran mesin.

Contoh ilustrasi gambar dapat dilihat pada gambar 4.4

6. TF-IDF memberi kita frekuensi kata dalam setiap dokumen atau mengganti data jadi number. Ini adalah rasio berapa

kali kata itu muncul dalam dokumen dibandingkan dengan jumlah total kata dalam dokumen itu. Itu meningkat seiring jumlah kemunculan kata itu di dalam dokumen meningkat. Setiap dokumen memiliki tf sendiri.

Contoh ilustrasi gambar dapat dilihat pada gambar 4.5





Figure 4.4: Bag Of Words

### 4.3.2 Praktek Program

1. Kali ini kita akan membuat data dummy dengan format csv di sini saya mengambil data dari web UCI Machine Learning

Repository dengan nama file Holiday.csv seperti pada gambar 4.6

Pada codingan 4.6 yaitu mengimport librari pandas dimana kita mengaliaskan praktek sebagai pandas. Pandas berguna untuk mengelola dataframe = matrix = tabel kemudian memanggil nama alias untuk membaca format csvnya.

2. Dari dataframe yang sudah ada sebelumnya kita akan membagi menjadi 2 yaitu 450 row pertama dan 50 row sisanya dapat dilihat pada gambar 4.7

Pada codingan 4.7 yaitu baris pertama dimana d\_train untuk membagi data training sebanyak 450 row dan pada

baris kedua dimana d\_test untuk data sisa atau data yang baru sebanyak 50 row.

3. Melakukan Vektorisasi dan Klasifikasi Data pada data Eminem pada gambar 4.8  
huda merupakan dataframe keseluruhan dari file csv yang sudah dimasukkan dengan jumlah 448 baris dan 5 kolom. nospam merupakan dataframe yang isinya hanya data yang bukan termasuk spam dengan inisial angka 0. Sedangkan

Kalimat 1		Kalimat 2	
Pagi nanti jangan lupa makan		Pagi nanti jangan pergi dulu	

Term	TF		IDF
	Kalimat 1	Kalimat 2	df
Pagi	1	1	2
nanti	1	1	2
jangan	1	1	2
lupa	1	0	1
makan	1	0	1
pergi	0	1	1
dulu	0	1	1

Figure 4.5: TF-IDF

spam merupakan dataframe yang isinya hanya data spam dengan inisial angka 1.

Pada gambar 4.9 merupakan hasil output content dimana terdapat 448 baris/data yang mempunyai 1602 kata-kata yang digunakan pada komentar yang ada di content tersebut.

Pada gambar 4.10 maksud dari outputannya merupakan dataframe kata-kata tadi yang berjumlah 1602 kata orang yang komen pada data eminem.

4. Mengklasifikasikan dari data vektorisasi dengan menggunakan klasifikasi svm(support vektorisasi machine) dapat dilihat pada gambar 4.11.

Jadi pada gambar 4.11 merupakan hasil dari memprediksi data score vektorisasi dengan svm menggunakan metode fit dimana digunakan untuk data training atau data pelatihannya saja.

5. Mengklasifikasikan dari data vektorisasi dengan menggunakan klasifikasi Decision Tree dapat dilihat pada gambar 4.12.

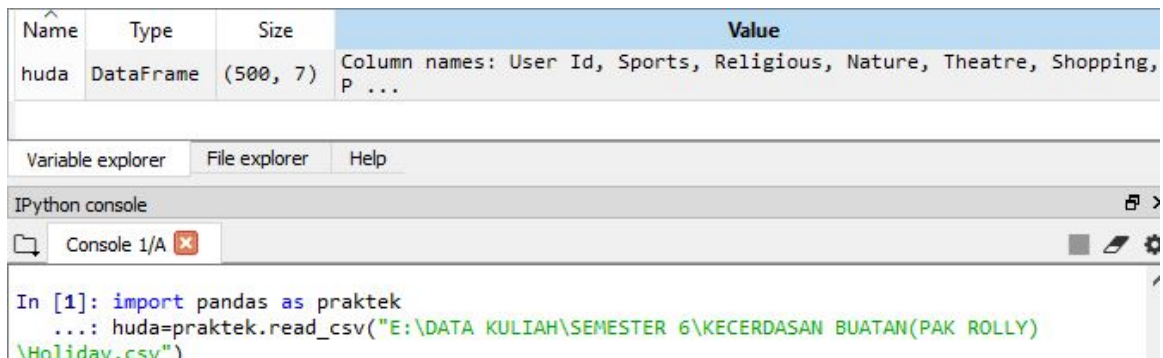


Figure 4.6: Data Dummy 500 Data

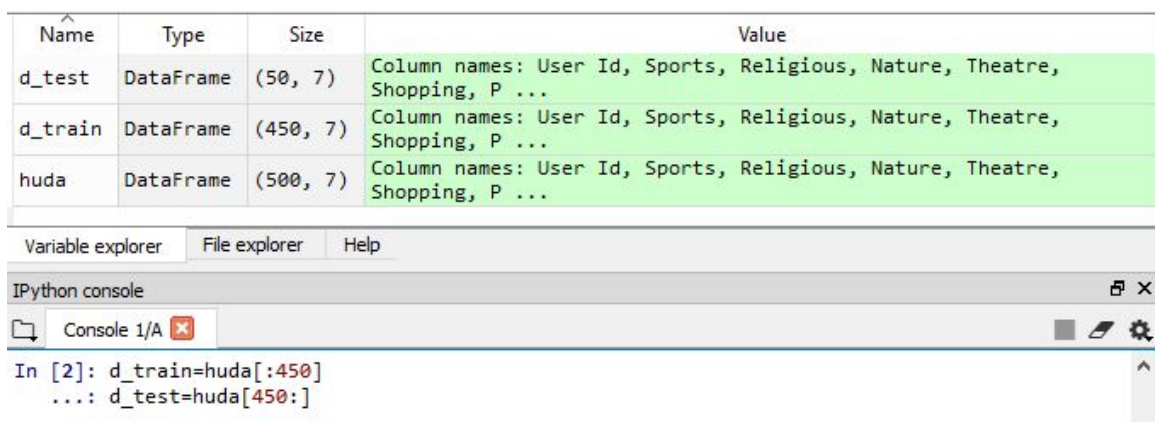


Figure 4.7: Membagi 2 Dataframe

Jadi pada gambar 4.12 merupakan hasil dari memprediksi data score vektorisasi dengan Decision Tree menggunakan metode fit dimana digunakan untuk data training atau data pelatihannya saja.

6. Mengeplot confusion matrix menggunakan matplotlib dapat dilihat pada gambar 4.13.

Pada gambar 4.13 sebelumnya kita harus mengimport matplotlibnya terlebih dahulu setelah itu di sini saya menggunakan numpy untuk mengeluarkan hasil plot confusion matrix pada matplotlibnya nantinya akan keluar normalisasi dari confusion matrix berupa data baris dan kolom.

7. Menjalankan program cross validation pada data vektorisasi dapat dilihat pada gambar 4.14.

huda	DataFrame	(448, 5)	Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS
nospam	DataFrame	(203, 5)	Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS
spam	DataFrame	(245, 5)	Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS

Figure 4.8: Vektorisasi dan Klasifikasi Data

```
<448x1602 sparse matrix of type '<class 'numpy.int64''>'
  with 7105 stored elements in Compressed Sparse Row format>
```

Figure 4.9: Data Content

Pada gambar 4.14 yang pertama yaitu memunculkan akurasi cross validation dari random forest pada data yang sudah di vektorisasi, yang kedua yaitu memunculkan akurasi cross validation dari decision tree pada data yang sudah di vektorisasi, dan yang ketiga yaitu memunculkan akurasi cross validation dari svm pada data yang sudah di vektorisasi.

8. Membuat program pengamatan komponen informasi pada data yang sudah di vektorisasi dapat dilihat pada gambar 4.15.

Pada gambar 4.15 merupakan hasil outputan yang mana max features di sana terdapat 9 data dari 10 yang sudah kita masukkan ke dalam codingan sebelumnya dan penomoran estimatornya merupakan data per 5 dari angka 40 sedangkan rata-rata akurasi kita tuliskan datanya mulai dari 0.9 sampai dengan 1. Titik-titik yang didalam tersebut merupakan data vektorisasi dari pengamatan komponen informasinya.

### 4.3.3 Penanganan Error

1. Pada gambar 4.16 merupakan ScreeShootan dari data yang eror.

2. `clfsvm = svm.SVC()`

```
F:\anaconda\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The def
"avoid this warning.", FutureWarning)
```

3. Solusi pemecah masalah eror tersebut dengan mengganti dan menambahkan codingan tersebut seperti berikut.

```
from sklearn import svm
clfsvm = svm.SVR(gamma='auto')
```

Name	Type	Size	Value
dk	list	1602	['00', '000', '047000', '09', '10', '100', '1000', '100877300245414', ...]

Figure 4.10: DataFrame Kata-kata Pada Content

```
In [20]: from sklearn import svm
...: clfsvm = svm.SVR(gamma='auto')
...: clfsvm.fit(huda_train_att, huda_train_label)
...: clfsvm.score(huda_test_att, huda_test_label)
Out[20]: 0.3360974270084298
```

Figure 4.11: Klasifikasi SVM Dari Data Vektorisasi

```
clfsvm.fit(huda_train_att, huda_train_label)
clfsvm.score(huda_test_att, huda_test_label)
```

```

In [19]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(huda_train_att, huda_train_label)
...: clftree.score(huda_test_att, huda_test_label)
Out[19]: 0.9324324324324325

```

Figure 4.12: Klasifikasi Decision Tree Dari Data Vektorisasi

```

In [22]: import numpy as np
...: np.set_printoptions(precision=2)
...: plot_confusion_matrix(cm, classes=huda, normalize=True)
...: plt.show()
Normalized confusion matrix
[[0.97 0.03]
 [0.06 0.94]]

```

Figure 4.13: Plot Confusion Matrix Menggunakan Matplotlib

```

In [23]: from sklearn.model_selection import cross_val_score
...: scores = cross_val_score(clf, huda_train_att, huda_train_label, cv=5)
...: # show average score and +/- two standard deviations away (covering 95% of scores)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
Accuracy: 0.95 (+/- 0.03)

In [24]: scorestree = cross_val_score(clftree, huda_train_att, huda_train_label, cv=5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(), scorestree.std() * 2))
Accuracy: 0.95 (+/- 0.05)

In [25]: scoressvm = cross_val_score(clfsvm, huda_train_att, huda_train_label, cv=5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(), scoressvm.std() * 2))
Accuracy: 0.30 (+/- 0.42)

```

Figure 4.14: Program Cross Validation Pada Data Vektorisasi

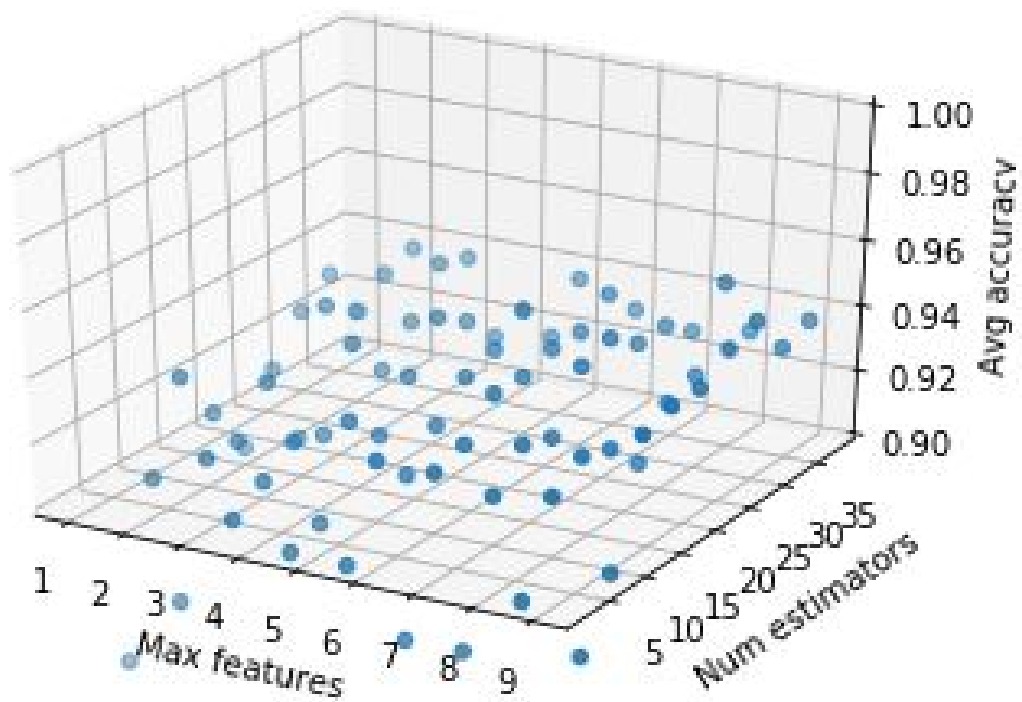


Figure 4.15: Program Pengamatan Komponen Informasi

```
In [28]: from sklearn import svm
...: clfsvm = svm.SVC()
...: clfsvm.fit(huda_train_att, huda_train_label)
...: clfsvm.score(huda_test_att, huda_test_label)
F:\anaconda\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The default value of
gamma will change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
"avoid this warning.", FutureWarning)
Out[28]: 0.5945945945945946
```

Figure 4.16: Error Pada Coding SVM

# Chapter 5

## Conclusion

brief of conclusion

### 5.1 Ahmad Syafrizal Huda/1164062

#### 5.1.1 Teori

1. Jelaskan Kenapa Kata-Kata harus dilakukan vektorisasi lengkapi dengan ilustrasi gambar.

Kata kata harus dilakukan vektorisasi dikarenakan untuk mengukur nilai kemunculan suatu kata yang sama dari sebuah kalimat sehingga kata-kata tersebut dapat di prediksi berapa kemunculanya. Atau juga di buatkan vektorisasi data yang digunakan untuk memprediksi bobot dari suatu kata misalkan mobil dan motor sama-sama kendaraan maka akan dibuat prediksi apakah kata tersebut akan muncul pada kalimat yang kira-kira memiliki bobot yang sama.

Untuk ilustrasinya dapat dilihat pada gambar 5.1



Figure 5.1: Ilustrasi Soal No. 1

2. Jelaskan Mengapa dimensi dari vektor dataset google bisa mencapai 300 lengkapi dengan ilustrasi gambar.



Dimensi dari vektor dataset google bisa mencapai 300 karena dimensi dari vektor digunakan untuk membandingkan bobot dari setiap kata, misalkan terdapat kata mobil dan motor pada dataset google tersebut setiap kata tersebut di buat dimensi vektor 300 untuk kata mobil dan 300 dimensi vektor juga untuk kata motor kemudian kata tersebut di bandingkan bobot kesamaan katanya maka akan muncul akurasi sekitar 70an persen kesamaan bobot dikarenakan kata mobil dan motor sama sama di gunakan sebagai kendaraan.

Untuk ilustrasinya dapat dilihat pada gambar 5.2

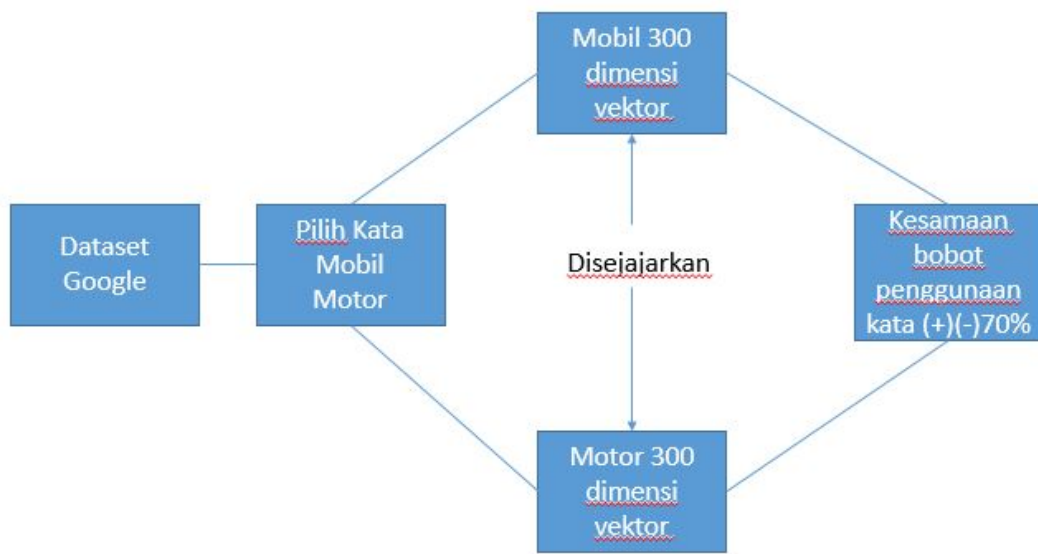


Figure 5.2: Ilustrasi Soal No. 2

3. Jelaskan Konsep vektorisasi untuk kata . dilengkapi dengan ilustrasi atau gambar.

Konesp vektorisasi untuk kata yaitu mengetahui kata tengah dari suatu kalimat utama dengan suatu kalimat contoh ( Jangan lupa like dan comment yah makasih ) kata tengah tersebut merupakan (dan) yang memiliki bobot sebagai kata tengah dari suatu kalimat atau bobot sebagai objek dari suatu kalimat. hal ini sangat berkaitan dengan dimensi vektor pada dataset google yang 300 tadi karena untuk mendapatkan nilai atau bobot dari kata tengah tersebut di dapatkan dari proses dimensi dari kata tersebut.

Untuk ilustrasinya dapat dilihat pada gambar 5.3



Figure 5.3: Ilustrasi Soal No. 3

4. Jelaskan Konsep vektorisasi untuk dokumen. dilengkapi dengan ilustrasi atau gambar.

Konsep vektorisasi untuk dokumen hampir sama seperti vektorisasi untuk kata hanya saja pemilihan kata utama atau kata tengah terdapat pada satu dokumen jadi mesin akan membuat dimensi vektor 300 untuk dokumen dan nanti kata tengahnya akan di sandingkan pada dokumen yang terdapat pada dokumen tersebut.

Untuk ilustrasinya dapat dilihat pada gambar 5.4



Figure 5.4: Ilustrasi Soal No. 4

5. Jelaskan apa mean dan standar deviasi, lengkapi dengan iludtrasi atau gambar.

Mean adalah nilai rata-rata dari suatu data. Mean disini merupakan petunjuk terhadap kata-kata yang di olah jika kata kata itu akurasiya tinggi berarti kata tersebut sering muncul begitu juga sebaliknya. Standar deviasi merupakan standar untuk menimbang kesalahan. Misalkan kita memperkirakan kedalaman dari dataset merupakan 2 atau 3 tapi pada kenyataanya merupakan 5 itu merupakan kesalahan tapi masih bisa dianggap wajar karena masih mendekati perkiraan awal.

Untuk ilustrasinya dapat dilihat pada gambar 5.5

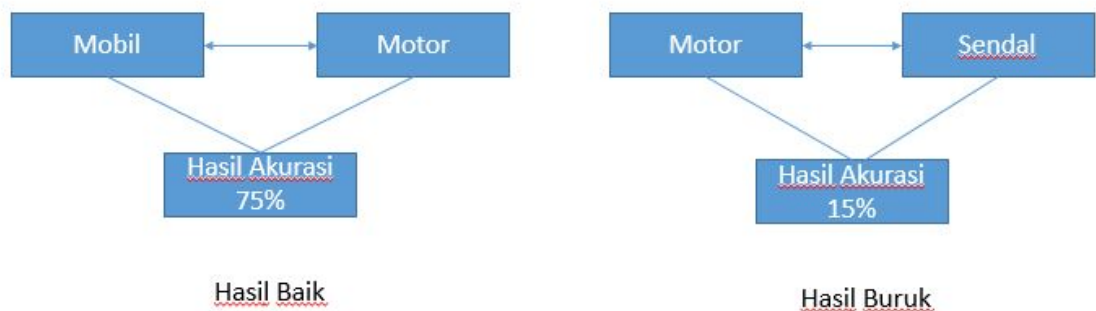


Figure 5.5: Ilustrasi Soal No. 5

6. Jelaskan Apa itu Skip-Gram sertakan contoh ilustrasi.

Skip-Gram yaitu dimana kata tengah menjadi acuan terhadap kata kata pelengkap dalam suatu kalimat.

Untuk ilustrasinya dapat dilihat pada gambar 5.6

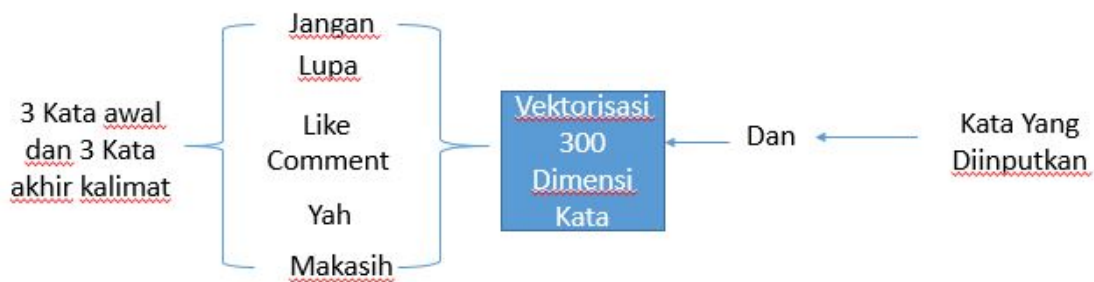


Figure 5.6: Ilustrasi Soal No. 6

### 5.1.2 Praktek Program

1. Cobalah dataset google, dan jelaskan vektor dari kata love, faith, fall, sick, clear, shine, bag, car, wash, motor, cycle dan cobalah untuk melakukan perbandingan similirati dari masing-masing kata tersebut. Jelaskan arti dari outputan similaritas.

Output source code dibawah akan memunculkan data vektor untuk kata love. bahwa vektor memiliki array sebanyak 300 dimensi. Hasil pada source code tersebut dapat dilihat pada gambar 5.7.

```
In [34]: gmodel['love']  
Out[34]:  
array([ 0.10302734, -0.15234375,  0.02587891,  0.16503906, -0.16503906,
```

Figure 5.7: Love

```
gmodel['love']
```

Output source code dibawah akan memunculkan data vektor untuk kata faith. bahwa vektor memiliki array sebanyak 300 dimensi. Hasil pada source code tersebut dapat dilihat pada gambar 5.8.

```
gmodel['faith']
```

```
In [35]: gmodel['faith']  
Out[35]:  
array([ 0.26367188, -0.04150391,  0.1953125 ,  0.13476562, -0.14648438,
```

Figure 5.8: Faith

Output source code dibawah akan memunculkan data vektor untuk kata fall. bahwa vektor memiliki array sebanyak 300 dimensi. Hasil pada source code tersebut dapat dilihat pada gambar 5.9.

```
gmodel['fall']
```

```
In [36]: gmodel['fall']  
Out[36]:  
array([-0.04272461,  0.10742188, -0.09277344,  0.16894531, -0.1328125 ,
```

Figure 5.9: Fall

Output source code dibawah akan memunculkan data vektor untuk kata sick. bahwa vektor memiliki array sebanyak 300 dimensi. Hasil pada source code tersebut dapat dilihat pada gambar 5.10.

```
gmodel['sick']
```

Output source code dibawah akan memunculkan data vektor untuk kata clear. bahwa vektor memiliki array sebanyak 300 dimensi. Hasil pada source code tersebut dapat dilihat pada gambar 5.11.

```
In [37]: gmodel['sick']
Out[37]:
array([ 1.82617188e-01,  1.49414062e-01, -4.05273438e-02,  1.64062500e-01,
```

Figure 5.10: Sick

```
In [38]: gmodel['clear']
Out[38]:
array([-2.44140625e-04, -1.02050781e-01, -1.49414062e-01, -4.24804688e-02,
```

Figure 5.11: Clear

```
gmodel['clear']
```

Output source code dibawah akan memunculkan data vektor untuk kata shine. bahwa vektor memiliki array sebanyak 300 dimensi. Hasil pada source code tersebut dapat dilihat pada gambar 5.12.

```
gmodel['shine']
```

```
In [39]: gmodel['shine']
Out[39]:
array([-0.12402344,  0.25976562, -0.15917969, -0.27734375,  0.30273438,
```

Figure 5.12: Shine

Output source code dibawah akan memunculkan data vektor untuk kata bag. bahwa vektor memiliki array sebanyak 300 dimensi. Hasil pada source code tersebut dapat dilihat pada gambar 5.13.

```
gmodel['bag']
```

```
In [40]: gmodel['bag']
Out[40]:
array([-0.03515625,  0.15234375, -0.12402344,  0.13378906, -0.11328125,
```

Figure 5.13: Bag

Output source code dibawah akan memunculkan data vektor untuk kata car. bahwa vektor memiliki array sebanyak 300 dimensi. Hasil pada source code tersebut dapat dilihat pada gambar 5.14.

```
In [41]: gmodel['car']  
Out[41]:  
array([ 0.13085938,  0.00842285,  0.03344727, -0.05883789,  0.04003906,
```

Figure 5.14: Car

```
gmodel['car']
```

Output source code dibawah akan memunculkan data vektor untuk kata wash. bahwa vektor memiliki array sebanyak 300 dimensi. Hasil pada source code tersebut dapat dilihat pada gambar 5.15.

```
gmodel['wash']
```

```
In [42]: gmodel['wash']  
Out[42]:  
array([ 9.46044922e-03,  1.41601562e-01, -5.46875000e-02,  1.34765625e-01,
```

Figure 5.15: Wash

Output source code dibawah akan memunculkan data vektor untuk kata motor. bahwa vektor memiliki array sebanyak 300 dimensi. Hasil pada source code tersebut dapat dilihat pada gambar 5.16.

```
gmodel['motor']
```

```
In [43]: gmodel['motor']  
Out[43]:  
array([ 5.73730469e-02,  1.50390625e-01, -4.61425781e-02, -1.32812500e-01,
```

Figure 5.16: Motor

Output source code dibawah akan memunculkan data vektor untuk kata cycle. bahwa vektor memiliki array sebanyak 300 dimensi. Hasil pada source code tersebut dapat dilihat pada gambar 5.17.

```
gmodel['cycle']
```

Pada source code dibawah menunjukkan hasil score perbandingan kata apakah kata motor dan cycle memiliki ke samaan atau tidak. Hasil pada source code tersebut dapat dilihat pada gambar 5.18.

```
In [44]: gmodel['cycle']
Out[44]:
array([ 0.04541016,  0.21679688, -0.02709961,  0.12353516, -0.20703125,
```

Figure 5.17: Cycle

```
In [45]: gmodel.similarity('motor', 'cycle')
Out[45]: 0.1794929675764453
```

Figure 5.18: Similariti Pada Kata Motor dan Cycle

```
gmodel.similarity('motor', 'cycle')
```

Pada source code dibawah menunjukkan hasil score perbandingan kata apakah kata wash dan motor memiliki ke samaan atau tidak. Hasil pada source code tersebut dapat dilihat pada gambar 5.19.

```
gmodel.similarity('wash', 'motor')
```

```
In [46]: gmodel.similarity('wash', 'motor')
Out[46]: 0.10280077965607967
```

Figure 5.19: Similariti Pada Kata Wash dan Motor

Untuk Motor dan Cycle hasilnya adalah 17%

Untuk Wash dan Motor hasilnya adalah 10%

Artinya Motor dan Cyle memang dalam kategori yang sama misalnya dalam kategori kata-kata yang disatukan/berpasangan. Mesin sudah mengetahui bahwa keduanya dapat dikategorikan sebagai sepasang kata.

2. Jelaskan dengan kata dan ilustrasi fungsi dari `extract_words` dan `PermuteSentences`.

`Extract_Words` merupakan function untuk menambahkan, menghilangkan atau menghapuskan, hal hal yang tidak penting atau tidak perlu di dalam teks. Pada gambar 5.20 berikut ini menggunakan function `extract_words` untuk

```
In [48]: import re
...: def extract_words(luarbiasa):
...:     luarbiasa = luarbiasa.lower()
...:     luarbiasa = re.sub(r'<[^>]+>', ' ', luarbiasa) #hapus tag html
...:     luarbiasa = re.sub(r'(\w)\'(\w)', ' ', luarbiasa) #hapus petik satu
...:     luarbiasa = re.sub(r'\W', ' ', luarbiasa) #hapus tanda baca
...:     luarbiasa = re.sub(r'\s+', ' ', luarbiasa) #hapus spasi yang berurutan
...:     return luarbiasa.split()
```

Figure 5.20: Extract\_Words

menghapus komen dengan python style , mencari data yang diinginkan, dan memberikan spasi pada teks.

PermuteSentences berfungsi untuk melakukan pengacakan data supaya memperoleh data yang teratur. Ini merupakan class yang digunakan untuk melakukan pengocokan secara acak pada data yang ada. Digunakan cara ini agar tidak terjadi kelebihan memori pada saat dijalankan. Hasilnya dapat dilihat pada gambar 5.21.

```
In [49]: import random
...: class PermuteSentences(object):
...:     def __init__(self, luarbiasas):
...:         self.luarbiasas = luarbiasas
...:
...:     def __iter__(self):
...:         shuffled = list(self.luarbiasas)
...:         random.shuffle(shuffled)
...:         for luarbiasa in shuffled:
...:             yield luarbiasa
```

Figure 5.21: Permute Sentences

3. Jelaskan fungsi dari librari gensim TaggedDocument dan Doc2Vec disertai praktek pemakaiannya.

Fungsi dari library gensim yaitu sebagai pemodelan topik tanpa pengawasan dan pemrosesan bahasa alami, atau bisa kita sebut dengan unsupervised. tagged document itu sendiri untuk memasukan kata-kata pada setiap dokumennya yang akan di vektorisasi. Fungsi dari doc2vec itu sendiri ialah untuk membandingkan



bobot data yang terdapat pada dokumen lainnya, apakah kata-kata didalamnya ada yang sama atau tidak. Ketika di running maka tidak terjadi apa-apa, seperti pada gambar 5.22

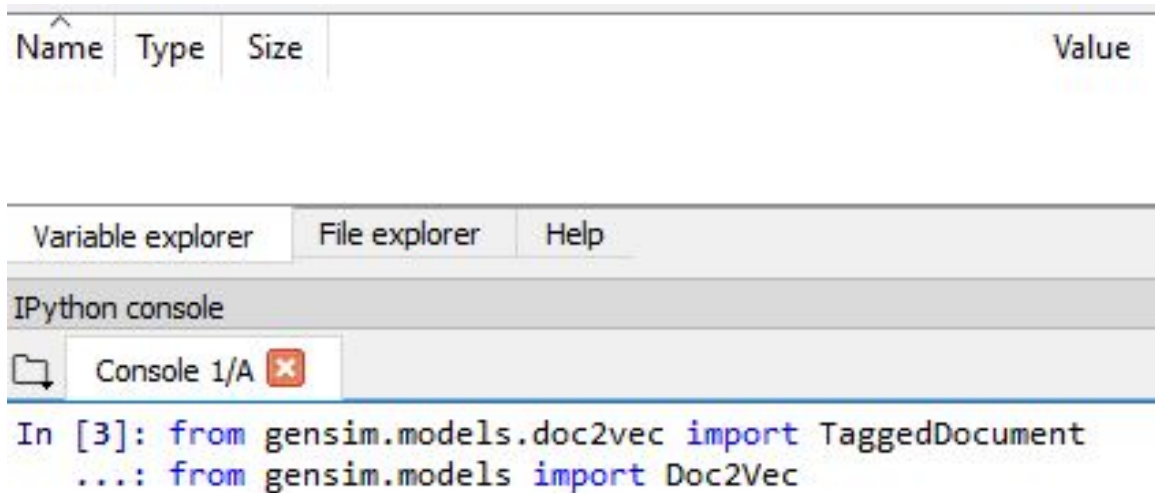


Figure 5.22: Gensim TaggedDocument dan Doc2vec

4. Jelaskan dengan kata dan praktek cara menambahkan data training dari file yang dimasukkan kepada variabel dalam rangka melatih model doc2vec.

Untuk menambahkan data training yaitu melakukan import library os, library os berfungsi untuk melakukan interaksi antara python dengan os laptop kita masing-masing, setelah itu kita buat variabel unsup sentences. Kemudian pilih direktori tempat data kita disimpan. Lalu menyortir data yang terdapat pada folder aclImdb dan membaca file tersebut dengan ekstensi .txt.

```
import os
unsup_sentences = []
for dirname in ["train/pos", "train/neg", "train/unsup", "test/pos", "test/neg"]:
    for fname in sorted(os.listdir("aclImdb/"+dirname)):
        if fname[-4:] == '.txt':
            with open("aclImdb/"+dirname+"/"+fname, encoding='UTF-8') as f:
                sent = f.read()
                words = extract_words(sent)
                unsup_sentences.append(TaggedDocument(words, [dirname+"/"+fname]))
```

Hasil dari code yang diatas ialah terdapatnya data training dengan jumlah 10000 data dari variabel `unsup_sentences` hasil running dari folder `aclImdb` dapat dilihat pada gambar 5.23

Name	Type	Size	Value
dirname	str	1	test/neg
fname	str	1	9_4.txt
sent	str	1	David Bryce's comments nearby are exceptionally well written and infor ...
unsup_sentences	list	100000	[TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument, Tagge ...
words	list	391	['david', 'bryc', 'comments', 'nearby', 'are', 'exceptionally', 'well' ...

Figure 5.23: Aclimbdb

```

for dirname in ["review_polarity/txt_sentoken/pos","review_polarity/txt_sentoken/neg"]:
    for fname in sorted(os.listdir(dirname)):
        if fname[-4:] == '.txt':
            with open(dirname+"/"+fname,encoding='UTF-8') as f:
                for i, sent in enumerate(f):
                    words = extract_words(sent)
                    unsup_sentences.append(TaggedDocument(words,["%s/%s-%d" % (dirname,fname,i)]))

```

Untuk code berikutnya akan menambahkan data training pada variabel `un-sup_sentences` sekitar 64.720 data, seperti pada gambar 5.24

Name	Type	Size	Value
dirname	str	1	txt_sentoken/neg
fname	str	1	cv999_14636.txt
i	int	1	24
sent	str	1	after watching _a_night_at_the_roxbury_ , you'll be left with exactly ...
unsup_sentences	list	164720	[TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument, Tagge ...
words	list	11	['after', 'watching', '_a_night_at_the_roxbury_', 'yo', 'l', 'be', 'le ...

Figure 5.24: Aclimbdb

```

with open("stanfordSentimentTreebank/original_rt_snippets.txt",encoding='UTF-8') as f:
    for i, sent in enumerate(f):

```

```
words = extract_words(sent)
unsup_sentences.append(TaggedDocument(words, ["rt-%d" % i]))
```

Untuk code berikutnya akan menambahkan data training pada variabel `unsup_sentences` sekitar 10.605 data, seperti pada gambar 5.25

Name	Type	Size	Value
dirname	str	1	txt_sentoken/neg
fname	str	1	cv999_14636.txt
i	int	1	10604
sent	str	1	Her fans walked out muttering words like ``horrible'' and ``terrible,' ...
unsup_sentences	list	175325	[TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument, Tagge ...
words	list	29	['her', 'fans', 'walked', 'out', 'muttering', 'words', 'like', 'horrib ...

Figure 5.25: Aclimbdb

5. Jelaskan dengan kata dan praktek kenapa harus dilakukan pengocokan dan pembersihan data.

Pengocokan data itu berguna untuk mengacak data supaya pada saat data di running bisa berjalan lebih baik dan hasil presentase akhirnya bisa lebih bagus. Sedangkan pembersihan data untuk memberikan ruang bagi ram komputer kita setelah melakukan running data sebanyak 3 juta lebih, agar lebih ringan saat proses selanjutnya. Hasil dari pengacakan data tidak ditampilkan pada spyder. Dan sebelumnya memori yang terpakai itu sekitar 87% , setelah dikosongkan jadi 71%. Untuk source codenya dapat dilihat sebagai berikut:

```
# Pengocokan data
mute = PermuteSentences(unsup_sentences)
# Pembersihan data
model.delete_temporary_training_data(keep_inference=True)
```

6. Jelaskan dengan kata dan praktek kenapa model harus di save dan kenapa temporari training harus dihapus.

Save data untuk menyimpan file hasil dari proses training data sebelumnya, model tersebut dilakukan penyimpanan untuk memberikan keringanan pada ram agar saat kita akan melakukan training lagi, model tersebut tinggal di muat saja tanpa harus melakukan tranning dari awal dan bisa menghemat waktu.

Sedangkan untuk delete temporary training data untuk menghapus data training yang sebelumnya sudah dilakukan dan disimpan, tujuannya memberikan keringanan pada ram. Karena setelah melakukan proses training, ram biasanya jadi berat untuk membaca sampai komputer kita jadi lemot. Untuk source codenya dapat dilihat sebagai berikut:

```
# Save data
model.save('ocean.d2v')
# Delete temporary data
model.delete_temporary_training_data(keep_inference=True)
```

7. Jalankan dengan kata dan praktek maksud dari infer code.

Infer vector yaitu membandingkan kata yang tercantum dengan vektor pada dokumen yang sudah di muat sebelumnya. Selain itu infer\_vector juga menghitung vektor dari kata yang dicantumkan pada model yang telah kita buat. Seharusnya kata yang dicantumkan itu lebih panjang lagi agar hasilnya bisa lebih baik lagi.

```
model.infer_vector(extract_words("I will go home"))
```

Pada source code tersebut menghasilkan keluaran seperti pada gambar 5.26.

```
In [14]: model.infer_vector(extract_words("I will go home"))
Out[14]:
array([-0.13122962, -0.2173184 , -0.14073701,  0.47829896,  0.10263892,
        -0.12006506, -0.02554635, -0.06321128,  0.01674332,  0.21312888,
        -0.03759272, -0.21068032,  0.08137176, -0.04404473,  0.23163871,
        -0.11791784, -0.02098055, -0.44984344, -0.21420991,  0.10826829,
        -0.24330835, -0.13386108,  0.05421483, -0.10329439, -0.16347748,
         0.03265174, -0.18829556,  0.14504528, -0.03915659, -0.39758494,
         0.05695166,  0.02784907, -0.15537408,  0.14412752, -0.0224928 ,
        -0.40351996, -0.11055487,  0.18106677,  0.09285883, -0.2614472 ,
         0.24014267,  0.15855208, -0.1287663 ,  0.53696984,  0.08572944,
         0.2602722 ,  0.3341717 ,  0.12752089,  0.15490048,  0.25898734,
        -0.23501003, -0.03285267], dtype=float32)
```

Figure 5.26: Infer Code

8. Jelaskan dengan praktek dan kata maksud dari cosine similarity.

Cosine similarity yaitu membandingkan vektorisasi data diantara kedua kata yang di masukkan, Jika hasil presentase dari kedua kata tersebut lebih dari 50 persen kemungkinan kata tersebut terdapat dalam 1 file. Tapi jika kurang dari 50 persen kata tersebut tidak terdapat dalam 1 file. Hasil yang didapatkan pada code tersebut hanya 0.4 persen itu dikarenakan kata pertama dan kedua tidak memiliki kesamaan vektorisasi dan tidak terdapat pada salah satu dokumen.

```
from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity(
    [model.infer_vector(extract_words("she going to school, after wash hand"))],
    [model.infer_vector(extract_words("Services sucks."))])
```

Pada source code tersebut menghasilkan keluaran seperti pada gambar 5.27.

```
In [15]: from sklearn.metrics.pairwise import cosine_similarity
...: cosine_similarity(
...:     [model.infer_vector(extract_words("she going to school, after wash hand"))],
...:     [model.infer_vector(extract_words("Services sucks."))])
Out[15]: array([[0.46642035]], dtype=float32)
```

Figure 5.27: Cosine Similarity

9. Jelaskan dengan praktek score dari cross validation masing-masing metode.

score dari cross validation yang mana mengecek data score KNN untuk menghasilkan typedata float64 dimana terdapat 5 data dari score yang typedatanya float64 tersebut. Dan disini menggunakan metode clf

```
scores = cross_val_score(clf, sentvecs, sentiments, cv=5)
np.mean(scores), np.std(scores)
```

Pada source code tersebut menghasilkan keluaran seperti pada gambar 5.28.

```
scores = cross_val_score(clfrf, sentvecs, sentiments, cv=5)
np.mean(scores), np.std(scores)
```

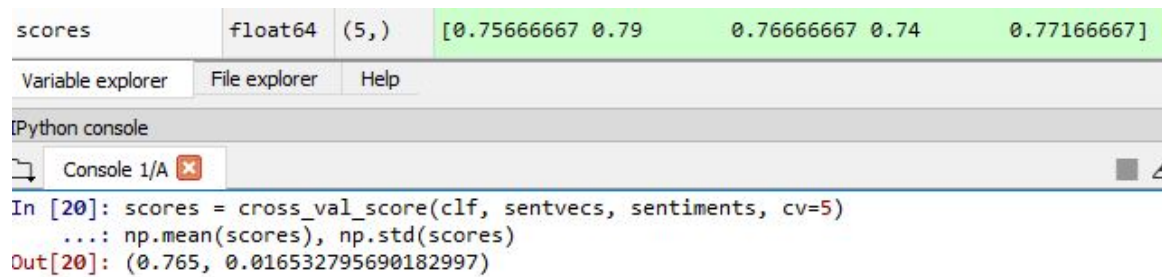


Figure 5.28: Cross Validation Metode 1

```
Out[21]: (0.7150000000000001, 0.02378141197564929)
```

Figure 5.29: Cross Validation Metode 2

Pada source code tersebut mengecek data score RF dengan menggunakan metode clrf yang menghasilkan keluaran seperti pada gambar 5.29.

```
from sklearn.pipeline import make_pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
pipeline = make_pipeline(CountVectorizer(), TfidfTransformer(), RandomForestClassifier())
scores = cross_val_score(pipeline, sentences, sentiments, cv=5)
np.mean(scores), np.std(scores)
```

Pada source code tersebut melakukan skoring dari vektorisasi, tfidf, dan rf lalu dibuat perbandingan yang menghasilkan keluaran seperti pada gambar 5.30.

```
Out[22]: (0.749, 0.01271918935222596)
```

Figure 5.30: Cross Validation Metode 3

# Chapter 6

## Discussion

Please tell more about conclusion and how to the next work of this study.

# Chapter 7

## Discussion

Please tell more about conclusion and how to the next work of this study.



# Chapter 8

## Discussion

Please tell more about conclusion and how to the next work of this study.

# Chapter 9

## Discussion

Please tell more about conclusion and how to the next work of this study.

# Chapter 10

## Discussion

Please tell more about conclusion and how to the next work of this study.

# Chapter 11

## Discussion

Please tell more about conclusion and how to the next work of this study.

# Chapter 12

## Discussion

Please tell more about conclusion and how to the next work of this study.

# Chapter 13

## Discussion

Please tell more about conclusion and how to the next work of this study.

# Chapter 14

## Discussion

Please tell more about conclusion and how to the next work of this study.

# Appendix A

## Form Penilaian Jurnal

gambar A.1 dan A.2 merupakan contoh bagaimana reviewer menilai jurnal kita.



NO	UNSUR	KETERANGAN	MAKS	KETERANGAN
1	Keefektifan Judul Artikel	Maksimal 12 (dua belas) kata dalam Bahasa Indonesia atau 10 (sepuluh) kata dalam Bahasa Inggris	2	a. Tidak lugas dan tidak ringkas (0) b. Kurang lugas dan kurang ringkas (1) c. Ringkas dan lugas (2)
2	Pencantuman Nama Penulis dan Lembaga Penulis		1	a. Tidak lengkap dan tidak konsisten (0) b. Lengkap tetapi tidak konsisten (0,5) c. Lengkap dan konsisten (1)
3	Abstrak	Dalam Bahasa Indonesia dan Bahasa Inggris yang baik, jumlah 150-200 kata. Isi terdiri dari latar belakang, metode, hasil, dan kesimpulan. Isi tertuang dengan kalimat yang jelas.	2	a. Tidak dalam Bahasa Indonesia dan Bahasa Inggris (0) b. Abstrak kurang jelas dan ringkas, atau hanya dalam Bahasa Inggris, atau dalam Bahasa Indonesia saja (1) c. Abstrak yang jelas dan ringkas dalam Bahasa Indonesia dan Bahasa Inggris (2)
4	Kata Kunci	Maksimal 5 kata kunci terpenting dalam paper	1	a. Tidak ada (0) b. Ada tetapi kurang mencerminkan konsep penting dalam artikel (0,5) c. Ada dan mencerminkan konsep penting dalam artikel (1)
5	Sistematika Pembahasan	Terdiri dari pendahuluan, tinjauan pustaka, metode penelitian, hasil dan pembahasan, kesimpulan dan saran, daftar pustaka	1	a. Tidak lengkap (0) b. Lengkap tetapi tidak sesuai sistematika (0,5) c. Lengkap dan bersistem (1)
6	Pemanfaatan Instrumen Pendukung	Pemanfaatan Instrumen Pendukung seperti gambar dan tabel	1	a. Tidak dimanfaatkan (0) b. Kurang informatif atau komplementer (0,5) c. Informatif dan komplementer (1)
7	Cara Pengacuan dan Pengutipan		1	a. Tidak baku (0) b. Kurang baku (0,5) c. Baku (1)
8	Penyusunan Daftar Pustaka	Penyusunan Daftar Pustaka	1	a. Tidak baku (0) b. Kurang baku (0,5) c. Baku (1)
9	Peristilahan dan Kebahasaan		2	a. Buruk (0) b. Baik (1) c. Cukup (2)
10	Makna Sumbangan bagi Kemajuan		4	a. Tidak ada (0) b. Kurang (1) c. Sedang (2) d. Cukup (3) e. Tinggi (4)

Figure A.1: Form nilai bagian 1.

11	Dampak Ilmiah		7	a. Tidak ada (0) b. Kurang (1) c. Sedang (3) d. Cukup (5) e. Besar (7)
12	Nisbah Sumber Acuan Primer berbanding Sumber lainnya	Sumber acuan yang langsung merujuk pada bidang ilmiah tertentu, sesuai topik penelitian dan sudah teruji.	3	a. < 40% (1) b. 40-80% (2) c. > 80% (3)
13	Derajat Kemutakhiran Pustaka Acuan	Derajat Kemutakhiran Pustaka Acuan	3	a. < 40% (1) b. 40-80% (2) c. > 80% (3)
14	Analisis dan Sintesis	Analisis dan Sintesis	4	a. Sedang (2) b. Cukup (3) c. Baik (4)
15	Penyimpulan	Sangat jelas relevasinya dengan latar belakang dan pembahasan, dirumuskan dengan singkat	3	a. Kurang (1) b. Cukup (2) c. Baik (3)
16	Unsur Plagiat		0	a. Tidak mengandung plagiat (0) b. Terdapat bagian-bagian yang merupakan plagiat (-5) c. Keseluruhannya merupakan plagiat (-20)
TOTAL			36	
Catatan : Nilai minimal untuk diterima 25				

Figure A.2: form nilai bagian 2.

# Appendix B

## FAQ

M : Kalo Intership II atau TA harus buat aplikasi ? D : Ga harus buat aplikasi tapi harus ngoding

M : Pa saya bingung mau ngapain, saya juga bingung mau presentasi apa? D : Makanya baca de, buka jurnal topik ‘ganteng’ nah kamu baca dulu sehari 5 kali ya, 4 hari udah 20 tuh. Bingung itu tanda kurang wawasan alias kurang baca.

M : Pa saya sudah cari jurnal terindeks scopus tapi ga nemu. D : Kamu punya mata de? coba dicolok dulu. Kamu udah lakuin apa aja? tolong di list laporkan ke grup Tingkat Akhir. Tinggal buka google scholar klik dari tahun 2014, cek nama jurnalnya di scimagojr.com beres.

M : Pa saya belum dapat tempat intership, jadi ga tau mau presentasi apa? D : kamu kok ga nyambung, yang dipresentasikan itu yang kamu baca bukan yang akan kamu lakukan.

M : Pa ini jurnal harus yang terindex scopus ga bisa yang lain ? D : Index scopus menandakan artikel tersebut dalam standar semantik yang mudah dipahami dan dibaca serta bukan artikel asal jadi. Jika diluar scopus biasanya lebih sukar untuk dibaca dan dipahami karena tidak adanya proses review yang baik dan benar terhadap artikel.

M : Pa saya tidak mengerti D : Coba lihat standar alasan

M : Pa saya bingung D : Coba lihat standar alasan

M : Pa saya sibuk D : Mbahmu....

M : Pa saya ganteng D : Ndasmu....

M : Pa saya kece D : wes karepmu lah....

Biasanya anda memiliki alasan tertentu jika menghadapi kendala saat proses bimbingan, disini saya akan melakukan standar alasan agar persepsi yang diterima sama dan tidak salah kaprah. Penggunaan kata alasan tersebut antara lain :

1. Tidak Mengerti : anda boleh menggunakan alasan ini jika anda sudah melakukan tahapan membaca dan meresumekan 15 jurnal. Sudah mencoba dan mempraktekkan teorinya dengan mencari di youtube dan google minimal 6 jam sehari selama 3 hari berturut-turut.

2. Bingung : anda boleh mengatakan alasan bingung setelah maksimal dalam berusaha menyelesaikan tugas bimbingan dari dosen(sudah dilakukan semua). Anda belum bisa mengatakan alasan bingung jika anda masih belum menyelesaikan tugas bimbingan dan poin nomor 1 diatas. Setelah anda menyelesaikan tugas bimbingan secara maksimal dan tahap 1 poin diatas, tapi anda masih tetap bingung maka anda boleh memakai alasan ini.

# Bibliography

- [1] Abdillah Baraja. Kecerdasan buatan tinjauan historikal. *Speed-Sentra Penelitian Engineering dan Edukasi*, 1(1), 2008.
- [2] Joshua Eckroth. *Python Artificial Intelligence Projects for Beginners: Get up and running with Artificial Intelligence using 8 smart and exciting AI applications*. Packt Publishing Ltd, 2018.
- [3] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.
- [4] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.