

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 penggerjaan 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

| | |
|--|-----------|
| 1 Mengenal Kecerdasan Buatan dan Scikit-Learn | 1 |
| 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 Persistence | 5 |
| 1.4.2.5 Conventions | 7 |
| 1.4.3 Penanganan error | 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 |
| 1.5 Cokro Edi Prawiro/1164069 | 12 |
| 1.5.1 Praktek teori penunjang | 12 |
| 1.5.2 Instalisasi | 18 |
| 1.6 Fathi Rabbani / 1164074 | 31 |
| 1.6.1 Teori | 31 |
| 1.6.2 Praktikum | 32 |
| 1.6.3 Penanganan Error | 37 |
| 2 Related Works | 49 |
| 2.1 Cokro Edi Prawiro/ 1164069 | 49 |
| 2.1.1 Teori | 49 |
| 2.1.2 Scikit-Learn /Cokro Edi Prawiro/1164069 | 51 |

| | | |
|----------|---|------------|
| 2.1.3 | Penanganan Error /Cokro Edi Prawiro/1164069 | 58 |
| 2.2 | Ahmad Syafrizal Huda/1164062 | 59 |
| 2.2.1 | Teori | 59 |
| 2.2.2 | Scikit-learn | 62 |
| 2.2.3 | Penanganan Eror | 66 |
| 2.3 | Fathi Rabbani / 1164074 | 68 |
| 2.3.1 | Teori | 68 |
| 2.3.2 | Praktek | 71 |
| 2.3.3 | Penanganan Error | 77 |
| 3 | Methods | 100 |
| 3.1 | Fathi Rabbani / 1164074 | 100 |
| 3.1.1 | Teori | 100 |
| 3.1.2 | Praktek | 102 |
| 3.1.3 | Error | 105 |
| 3.2 | Cokro Edi Prawiro / 1164069 | 105 |
| 3.2.1 | Teori | 105 |
| 3.2.2 | Praktikum | 119 |
| 3.2.3 | Penanganan Error / cokro | 122 |
| 3.3 | Ahmad Syafrizal Huda / 1164062 | 131 |
| 3.3.1 | Teori | 131 |
| 3.3.2 | Praktek Program | 134 |
| 3.3.3 | Penanganan Eror | 137 |
| 4 | Experiment and Result | 155 |
| 4.1 | Experiment | 155 |
| 4.2 | Result | 155 |
| 4.3 | Cokro Edi Prawiro / 1164069 | 155 |
| 4.3.1 | Teori | 155 |
| 4.3.2 | Praktek Program | 157 |
| 4.3.3 | Penanganan Error | 163 |
| 4.4 | Fathi Rabbani / 1164074 | 163 |
| 4.4.1 | Teori | 163 |
| 4.4.2 | Praktek | 176 |
| 4.4.3 | Error | 182 |

| | |
|---|------------|
| 5 Conclusion | 187 |
| 5.1 Cokro Edi Prawiro / 1164069 | 187 |
| 5.1.1 Teori | 187 |
| 5.1.2 Praktikum | 190 |
| 5.1.3 Penanganan Error | 202 |
| 5.2 Fathi Rabbani / 1164074 | 202 |
| 5.2.1 Teori | 202 |
| 5.2.2 Praktikum | 204 |
| 6 Discussion | 216 |
| 6.1 Cokro Edi Prawiro / 1164069 | 216 |
| 6.1.1 Teori | 216 |
| 6.2 Fathi Rabbani / 1164074 | 220 |
| 6.2.1 Teori | 220 |
| 7 Discussion | 225 |
| 8 Discussion | 226 |
| 9 Discussion | 227 |
| 10 Discussion | 228 |
| 11 Discussion | 229 |
| 12 Discussion | 230 |
| 13 Discussion | 231 |
| 14 Discussion | 232 |
| A Form Penilaian Jurnal | 233 |
| B FAQ | 236 |
| Bibliography | 238 |

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. | 13 |
| 1.5 | Langkah pertama instalasi anaconda. | 13 |
| 1.6 | Langkah kedua instalasi anaconda. | 14 |
| 1.7 | Langkah ketiga instalasi anaconda. | 15 |
| 1.8 | Langkah terakhir instalasi anaconda. | 16 |
| 1.9 | Langkah pertama instalasi scikit pada CMD. | 16 |
| 1.10 | Langkah kedua instalasi scikit pada CMD. | 17 |
| 1.11 | Langkah ketiga instalasi scikit pada CMD. | 17 |
| 1.12 | Langkah compile code pada python anaconda. | 18 |
| 1.13 | Hasil Tampilan 1. | 18 |
| 1.14 | Hasil Tampilan 2. | 19 |
| 1.15 | Hasil Tampilan 3. | 19 |
| 1.16 | Hasil Tampilan 4. | 20 |
| 1.17 | Hasil Tampilan 5. | 20 |
| 1.18 | Hasil Tampilan 6. | 21 |
| 1.19 | Hasil Tampilan 7. | 21 |
| 1.20 | Hasil Tampilan 8. | 22 |
| 1.21 | Hasil Tampilan 9. | 22 |
| 1.22 | Hasil Tampilan 10. | 23 |
| 1.23 | Hasil Tampilan 11. | 23 |
| 1.24 | Hasil Tampilan 12. | 24 |
| 1.25 | Hasil Tampilan 13. | 24 |
| 1.26 | Hasil Tampilan 14. | 25 |
| 1.27 | Hasil Tampilan 15. | 25 |
| 1.28 | Hasil Tampilan 16. | 26 |

| | |
|--|----|
| 1.29 Hasil Tampilan 17 | 26 |
| 1.30 Hasil Tampilan 18. | 27 |
| 1.31 Hasil Tampilan 19. | 27 |
| 1.32 Hasil Tampilan 20. | 28 |
| 1.33 Hasil Tampilan 21. | 28 |
| 1.34 Hasil Tampilan 22. | 29 |
| 1.35 Tampilan website Scikit 1. | 29 |
| 1.36 Tampilan website Scikit 2. | 30 |
| 1.37 setelah membuka data instalasi klik next | 39 |
| 1.38 pilih i agree | 39 |
| 1.39 pilih instalasi Just Me | 40 |
| 1.40 langsung saja next | 40 |
| 1.41 cek kedua pilihan tersebut | 41 |
| 1.42 proses Instalasi | 41 |
| 1.43 klik next | 42 |
| 1.44 selesai instalasi anaconda | 42 |
| 1.45 Instalasi SCIKIT dengan menggunakan anaconda | 43 |
| 1.46 Konfirmasi Instalasi | 43 |
| 1.47 hasil dari instalasi SCIKIT | 44 |
| 1.48 data variable explorer | 44 |
| 1.49 code example dataset yang digunakan | 44 |
| 1.50 data hasil dari code example dataset yang digunakan | 44 |
| 1.51 Tampilan Versi Python dan Anaconda | 45 |
| 1.52 Instalisasi Library Sikic. | 45 |
| 1.53 Instalasi Library Sikic Melalui Conda | 46 |
| 1.54 Console Python Include Anaconda | 46 |
| 1.55 Contoh Codingan Dataset | 46 |
| 1.56 Error Coding 1 | 47 |
| 1.57 Error Coding 2 | 47 |
| 1.58 Codingan Solusi Untuk Error digits | 47 |
| 1.59 Codingan Solusi Untuk Error Joblib | 47 |
| 1.60 Hasil Solusi Error Joblib | 48 |
| 1.61 Error Type data, yang harus digunakan number sedangkan isinya 'SCALE' pada gamma | 48 |
| 1.62 Error no Module found, modul yang dicari tidak ditemukan atau tidak ada 'JOBLIB' | 48 |

| | | |
|------|--|----|
| 2.1 | Source Code Load Dataset | 52 |
| 2.2 | Hasil Load Dataset | 52 |
| 2.3 | memberikan nilai satu atau nol | 53 |
| 2.4 | hasil dari memberikan nilai nol dan satu | 53 |
| 2.5 | Penambahan nilai numerik | 54 |
| 2.6 | hasil Penambahan nilai numerik | 54 |
| 2.7 | penentuan data training dan data testing | 54 |
| 2.8 | Hasil 1 penentuan data training dan data testing | 55 |
| 2.9 | hasil 2 penentuan data training dan data testing | 55 |
| 2.10 | memberikan nilai pada pohon keputusan | 56 |
| 2.11 | hasil memberikan nilai pada pohon keputusan | 56 |
| 2.12 | pembuatan diagram pohon keputusan | 57 |
| 2.13 | hasil pembuatan pohon keputusan | 57 |
| 2.14 | coding save | 58 |
| 2.15 | hasil coding save | 58 |
| 2.16 | Contoh Binary Classification | 59 |
| 2.17 | hasil coding save | 59 |
| 2.18 | Akurasi perhitungan pohon keputusan | 60 |
| 2.19 | hasil Akurasi perhitungan pohon keputusan | 60 |
| 2.20 | Contoh Binary Classification | 61 |
| 2.21 | hasil Akurasi perhitungan pohon keputusan | 61 |
| 2.22 | Contoh Binary Classification | 62 |
| 2.23 | hasil Akurasi perhitungan pohon keputusan | 63 |
| 2.24 | codingan pembuatan grafik | 64 |
| 2.25 | hasil grafik | 65 |
| 2.26 | Scrensot error | 66 |
| 2.27 | Proses instalasi graphviz | 67 |
| 2.28 | Proses instalasi graphviz di user | 68 |
| 2.29 | Path Komputer | 69 |
| 2.30 | Memasukan Direktori Ke Path | 70 |
| 2.31 | Hasil Codingan No 1. | 71 |
| 2.32 | Hasil Codingan No 2. | 71 |
| 2.33 | Hasil Codingan No 3. | 72 |
| 2.34 | Hasil Codingan No 4. | 73 |
| 2.35 | Hasil Codingan No 5. | 73 |
| 2.36 | Hasil Codingan No 6. | 74 |

| | |
|--|----|
| 2.37 Hasil Codingan No 7 | 74 |
| 2.38 Hasil Codingan No 8 | 75 |
| 2.39 Hasil Codingan No 9 | 75 |
| 2.40 Hasil Codingan No 10 | 76 |
| 2.41 Hasil Codingan No 11 | 77 |
| 2.42 Hasil Codingan No 12 | 78 |
| 2.43 Hasil Gambar Eror No 6 | 78 |
| 2.44 Hasil Gambar Penanganan Eror No 6 | 79 |
| 2.45 Binary Classification. | 80 |
| 2.46 Supervised Learning. | 81 |
| 2.47 Unsupervised Learning. | 81 |
| 2.48 Clustering. | 81 |
| 2.49 Evaluasi dan Akurasi. | 82 |
| 2.50 K-fold Cross Validation. | 82 |
| 2.51 Decision Tree. | 83 |
| 2.52 Gain. | 83 |
| 2.53 Contoh Binary Classification | 84 |
| 2.54 Ilustrasi Suervised Learning | 85 |
| 2.55 Ilustrasi Unsuervised Learning | 86 |
| 2.56 Ilustrasi Clustering | 86 |
| 2.57 Ilustrasi Evaluasi | 87 |
| 2.58 Ilustrasi Confusion Matrix | 88 |
| 2.59 Ilustrasi K-Fold | 89 |
| 2.60 Ilustrasi Decision Tree | 89 |
| 2.61 Contoh Ilustrasi Information Gain. | 90 |
| 2.62 Contoh Penggunaan Binary Classification | 90 |
| 2.63 Contoh Penggunaan Supervised Learning | 91 |
| 2.64 Contoh Penggunaan Unsupervised Learning | 91 |
| 2.65 Contoh Penggunaan Clustering | 91 |
| 2.66 Contoh Penggunaan Evaluasi dan Akurasi | 92 |
| 2.67 Contoh Matrix Confusion | 92 |
| 2.68 Contoh Matrix Confusion | 92 |
| 2.69 Contoh Penggunaan K Fold Cross Validation | 93 |
| 2.70 Contoh Penggunaan Decision Tree | 93 |
| 2.71 Contoh Penggunaan Information Gain | 94 |
| 2.72 code 1 hasil | 94 |

| | | |
|------|--|-----|
| 2.73 | code 2 hasil | 94 |
| 2.74 | code 3 hasil | 94 |
| 2.75 | code 4 hasil | 95 |
| 2.76 | code 5 hasil | 95 |
| 2.77 | code 6 hasil | 95 |
| 2.78 | code 7 hasil | 95 |
| 2.79 | code 8 hasil | 95 |
| 2.80 | code 9 hasil | 95 |
| 2.81 | code 10 hasil | 96 |
| 2.82 | code 11 hasil | 97 |
| 2.83 | code 12 hasil | 98 |
| 2.84 | Error Path | 98 |
| 2.85 | Fix Error | 99 |
| 3.1 | Random Forest | 106 |
| 3.2 | Hasil dari membaca data dengan Confusion Matriks | 107 |
| 3.3 | Voting Random Forest | 108 |
| 3.4 | Pandas Implementasi | 108 |
| 3.5 | Numpy Implementasi | 109 |
| 3.6 | Numpy Implementasi | 109 |
| 3.7 | Numpy Implementasi | 110 |
| 3.8 | Matplotlib Implementasi | 111 |
| 3.9 | Random Forest Classifier | 112 |
| 3.10 | Confusion Matrix | 112 |
| 3.11 | Decision Tree | 112 |
| 3.12 | Support Vector Machine | 113 |
| 3.13 | Cross Validation data Random Forest | 113 |
| 3.14 | Cross Validation data Decision Tree | 113 |
| 3.15 | Cross Validation data SVM | 113 |
| 3.16 | Pengamatan Akurasi data dengan Cross Validation | 114 |
| 3.17 | Pengamatan Akurasi data dengan Matplotlib | 115 |
| 3.18 | Cross Validation data Random Forest | 116 |
| 3.19 | Cross Validation data Random Forest | 116 |
| 3.20 | Hasil Confusion Matrix | 117 |
| 3.21 | Hasil Confusion Matrix | 118 |
| 3.22 | Random Forest | 123 |

| | |
|---|-----|
| 3.23 Ilustrasi Voting | 123 |
| 3.24 Ilustrasi Confusion matrix | 124 |
| 3.25 Contoh aplikasi sederhana pandas | 124 |
| 3.26 Hasil aplikasi sederhana pandas | 124 |
| 3.27 Contoh aplikasi sederhana numpy | 125 |
| 3.28 Contoh aplikasi sederhana matplotlib | 125 |
| 3.29 Hasil aplikasi sederhana matplotlib | 126 |
| 3.30 Hasil aplikasi random forest | 126 |
| 3.31 Hasil aplikasi confusion matrix | 127 |
| 3.32 Hasil aplikasi SVM dan Decision tree | 127 |
| 3.33 Hasil aplikasi SVM dan Decision tree | 127 |
| 3.34 Hasil aplikasi cross validation | 128 |
| 3.35 Hasil aplikasi cross validation | 128 |
| 3.36 Error Svm | 129 |
| 3.37 Error Svm | 129 |
| 3.38 Error Svm | 130 |
| 3.39 Error Svm | 130 |
| 3.40 Merubah data training | 131 |
| 3.41 Hasil Solusi | 131 |
| 3.42 Random Forest | 139 |
| 3.43 Hasil Dataset | 139 |
| 3.44 Confusion Matrix | 140 |
| 3.45 Voting | 141 |
| 3.46 Aplikasi Menggunakan Pandas | 141 |
| 3.47 Aplikasi Menggunakan Numpy | 142 |
| 3.48 Aplikasi Menggunakan Matplotlib | 142 |
| 3.49 Hasil 1 Random Forest | 142 |
| 3.50 Hasil 2 Random Forest | 143 |
| 3.51 Hasil 3 Random Forest | 143 |
| 3.52 Hasil 4 Random Forest | 143 |
| 3.53 Hasil 5 Random Forest | 144 |
| 3.54 Hasil 6 Random Forest | 144 |
| 3.55 Hasil 7 Random Forest | 144 |
| 3.56 Hasil 8 Random Forest | 145 |
| 3.57 Hasil 9 Random Forest | 145 |
| 3.58 Hasil 10 Random Forest | 145 |

| | |
|--|-----|
| 3.59 Hasil 11 Random Forest | 146 |
| 3.60 Hasil 11 Random Forest | 146 |
| 3.61 Hasil 12 Random Forest | 147 |
| 3.62 Hasil 13 Random Forest | 147 |
| 3.63 Hasil 14 Random Forest | 147 |
| 3.64 Hasil 15 Random Forest | 148 |
| 3.65 Hasil 16 Random Forest | 148 |
| 3.66 Hasil 17 Random Forest | 148 |
| 3.67 Hasil 1 Confusion Matrix | 148 |
| 3.68 Hasil 2 Confusion Matrix | 148 |
| 3.69 Hasil 3 Confusion Matrix | 149 |
| 3.70 Hasil 4 Confusion Matrix | 150 |
| 3.71 Hasil 5 Confusion Matrix | 151 |
| 3.72 Hasil 1 Klasifikasi SVM dan Decision Tree | 152 |
| 3.73 Hasil 2 Klasifikasi SVM dan Decision Tree | 152 |
| 3.74 Hasil 1 Cross Validation | 152 |
| 3.75 Hasil 2 Cross Validation | 152 |
| 3.76 Hasil 3 Cross Validation | 153 |
| 3.77 Hasil 1 Pengamatan Komponen Informasi | 153 |
| 3.78 Hasil 2 Pengamatan Komponen Informasi | 154 |
| 3.79 Eror | 154 |
| | |
| 4.1 Ilustrasi clasifikasi Teks | 164 |
| 4.2 Ilustrasi Bunga Tidak bisa dibaca di Mesin Learning | 164 |
| 4.3 contoh hasil pembelajaran text di youtube | 165 |
| 4.4 Ilustrasi bag of words | 165 |
| 4.5 Rumus TF-IDF | 166 |
| 4.6 Hasil Perhitungan TF-IDF | 167 |
| 4.7 Hasil Perhitungan TF-IDF | 168 |
| 4.8 Hasil running data vektorisasi | 169 |
| 4.9 Hasil running klasifikasi svm menggunakan data vektorisasi | 169 |
| 4.10 Hasil running klasifikasi decision treei | 169 |
| 4.11 Hasil running cnfusion matrix | 170 |
| 4.12 Hasil Cross Validationi | 170 |
| 4.13 pengulangan Isi Grafik | 171 |
| 4.14 Grafik 3D | 172 |

| | | |
|------|--|-----|
| 4.15 | Gambar Warning | 172 |
| 4.16 | Hasil penanganan Error | 172 |
| 4.17 | Klasifikasi Teks | 173 |
| 4.18 | Klasifikasi Bunga | 173 |
| 4.19 | Data Youtube | 174 |
| 4.20 | Bags of Word | 175 |
| 4.21 | TF-IDF | 176 |
| 4.22 | Decision Tree di Vektorisasikan | 183 |
| 4.23 | klasifikasi SVM | 183 |
| 4.24 | klasifikasi Tree | 184 |
| 4.25 | Confusion matrix | 184 |
| 4.26 | matplotlib | 184 |
| 4.27 | Cross validasi | 184 |
| 4.28 | Cross validasi pada tree | 184 |
| 4.29 | Cross validasi pada SVM | 185 |
| 4.30 | pengamatan program | 185 |
| 4.31 | Error | 186 |
| 5.1 | Ilustrasi Vektorisasi Kata-Kata | 187 |
| 5.2 | Ilustrasi Kenapa dimensi vektor pada datasets google haris 300 | 188 |
| 5.3 | Ilustrasi konsep vektorisasi untuk kata | 189 |
| 5.4 | Ilustrasi konsep vektorisasi untuk dokumen | 189 |
| 5.5 | Ilustrasi Penggunaan Mean | 190 |
| 5.6 | Ilustrasi Skip-Gram | 190 |
| 5.7 | Import Gensim dan membuat model gmodel | 190 |
| 5.8 | Hasil Matrix Clear | 191 |
| 5.9 | Hasil Matrix Shine | 191 |
| 5.10 | Hasil Matrix bag | 191 |
| 5.11 | Hasil Matrix Car | 192 |
| 5.12 | Hasil Matrix Wash | 192 |
| 5.13 | Hasil Matrix Motor | 192 |
| 5.14 | Hasil Matrix Cycle | 193 |
| 5.15 | Hasil Matrix love | 193 |
| 5.16 | Hasil Matrix faith | 193 |
| 5.17 | Hasil Matrix Fall | 194 |
| 5.18 | Hasil Matrix Sick | 194 |

| | |
|--|-----|
| 5.19 hasil dari lima similaritas | 194 |
| 5.20 hasil dari ekstrak_words dan permute Sentence | 195 |
| 5.21 Hasil dari penggunaan library gensim , TaggetDocument dan Doc2Vec | 195 |
| 5.22 Codingan untuk memasukan dokumen pelatihan doc2vec | 196 |
| 5.23 hasil running codingan ke 1 | 196 |
| 5.24 Hasil insert data doc2vec dari codingan pertama | 196 |
| 5.25 hasil running codingan ke 2 | 197 |
| 5.26 Hasil insert data doc2vec dari codingan ke dua | 197 |
| 5.27 hasil running codingan ke 3 | 197 |
| 5.28 Hasil insert data doc2vec dari codingan ke tiga | 197 |
| 5.29 Membuat random dan membuat class PermuteSentence | 198 |
| 5.30 Membuat variabel mute | 198 |
| 5.31 Code Pembersihan | 198 |
| 5.32 Membuat variabel mute | 199 |
| 5.33 Hasil save data vektorisasi | 199 |
| 5.34 Code untuk inver_code | 199 |
| 5.35 Hasil dari inver code | 199 |
| 5.36 Code untuk consine_simirarity | 200 |
| 5.37 Hasil dariconsine_simirarity | 200 |
| 5.38 Code untuk Cros Validation | 201 |
| 5.39 Hasil dari perhitungan KNeighborsClasifier | 201 |
| 5.40 Hasil dari perhitungan RandomForestClasifier | 201 |
| 5.41 Hasil dari perhitungan Cross Validation 1 | 201 |
| 5.42 Hasil dari perhitungan Cross Validation 2 | 201 |
| 5.43 Ilustrasi Vektorisasi Kata | 202 |
| 5.44 Ilustrasi Google dataset | 202 |
| 5.45 Ilustrasi Concept of Vectorizer on Words | 203 |
| 5.46 Ilustrasi Concept of Vectorizer on Document | 203 |
| 5.47 Ilustrasi Mean and Deviation Standart | 204 |
| 5.48 Ilustrasi skip-gram | 204 |
| 5.49 Ilustrasi import gensim dan olah data GoogleNews-vector | 205 |
| 5.50 Ilustrasi hasil olah data LOVE pada GoogleNews-vector | 205 |
| 5.51 Ilustrasi hasil olah data FAITH pada GoogleNews-vector | 205 |
| 5.52 Ilustrasi hasil olah data FALL pada GoogleNews-vector | 205 |
| 5.53 Ilustrasi hasil olah data SICK pada GoogleNews-vector | 206 |
| 5.54 Ilustrasi hasil olah data CLEAR pada GoogleNews-vector | 206 |

| | | |
|------|--|-----|
| 5.55 | Ilustrasi hasil olah data SHINE pada GoogleNews-vector | 206 |
| 5.56 | Ilustrasi hasil olah data BAG pada GoogleNews-vector | 206 |
| 5.57 | Ilustrasi hasil olah data CAR pada GoogleNews-vector | 207 |
| 5.58 | Ilustrasi hasil olah data WASH pada GoogleNews-vector | 207 |
| 5.59 | Ilustrasi hasil olah data MOTOR pada GoogleNews-vector | 207 |
| 5.60 | Ilustrasi hasil olah data CYCLE pada GoogleNews-vector | 208 |
| 5.61 | Ilustrasi hasil olah data pada GoogleNews-vector menggunakan SIMILARITY | 208 |
| 5.62 | Ilustrasi hasil olah data pada GoogleNews-vector menggunakan extract_words dan PermuteSentences | 209 |
| 5.63 | Ilustrasi TaggedDocument dan Doc2Vec | 209 |
| 5.64 | Ilustrasi data code praktek data training | 210 |
| 5.65 | Ilustrasi data code praktek data training | 210 |
| 5.66 | Ilustrasi data code praktek data training | 210 |
| 5.67 | Ilustrasi data code praktek data training | 211 |
| 5.68 | Ilustrasi data code praktek data training | 211 |
| 5.69 | Ilustrasi data code praktek data training | 211 |
| 5.70 | Ilustrasi data code praktek data training | 211 |
| 5.71 | Ilustrasi Shuffled dan Randomisasi data | 212 |
| 5.72 | Ilustrasi pembuatan variable muter untuk memuat data unsup_sentences | 212 |
| 5.73 | Ilustrasi code untuk membersihkan data memory | 212 |
| 5.74 | Ilustrasi data code save data | 213 |
| 5.75 | Ilustrasi hasil file simpan | 213 |
| 5.76 | Ilustrasi code dan hasil infer_vector | 213 |
| 5.77 | Ilustrasi code dan hasil penggunaan cosine_similarity | 214 |
| 5.78 | Ilustrasi code dan hasil penggunaan cosine_similarity | 214 |
| 5.79 | Ilustrasi memasukan code import library | 214 |
| 5.80 | Ilustrasi perhitungan data KNeighborsClassifier dengan cross validasi | 214 |
| 5.81 | Ilustrasi perhitungan data RandomForestClassifier dengan cross validasi | 215 |
| 5.82 | Ilustrasi perhitungan data Cross Validasi untuk nilai keseluruhan dari KNeighborsClassifier, RandomForestClassifier dan Vectorizer | 215 |
| 5.83 | Ilustrasi perhitungan data hasil persenan yang diakumulasi | 215 |
| 6.1 | Ilustrasi gambar metode MFCC | 216 |
| 6.2 | Ilustrasi Konsep dasar neural network | 217 |
| 6.3 | Ilustrasi Konsep pembobotan pada neural network | 217 |

| | | |
|------|---|-----|
| 6.4 | Gambar yang dibaca hasil plotnya | 218 |
| 6.5 | Ilustrasi Cara Membaca Hasil Plot | 218 |
| 6.6 | Ilustrasi Konsep one-hot encoding | 219 |
| 6.7 | Ilustrasi np.unique | 219 |
| 6.8 | Ilustrasi to_categorical | 219 |
| 6.9 | Ilustrasi Konsep pembobotan pada neural network | 220 |
| 6.10 | Ilustrasi MFCC | 220 |
| 6.11 | Ilustrasi Neural Network | 221 |
| 6.12 | Ilustrasi pembobotan Neural Network | 221 |
| 6.13 | Ilustrasi fungsi aktifasi Neural Network | 222 |
| 6.14 | Ilustrasi Membaca nilai Plot dari MFCC | 222 |
| 6.15 | Ilustrasi One-hot Encoding | 223 |
| 6.16 | Ilustrasi np.unique | 223 |
| 6.17 | Ilustrasi to_categorical | 223 |
| 6.18 | Ilustrasi fungsi Sequential | 224 |
| A.1 | Form nilai bagian 1. | 234 |
| A.2 | form nilai bagian 2. | 235 |

Chapter 1

Mengenal Kecerdasan Buatan dan Scikit-Learn

Buku umum yang digunakan adalah [5] 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 plagiatis[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)

iiiiiii HEAD

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 mengelompokan suatu data ke data yang sudah ada. Sedangkan unsupervised learning tidak memiliki data latih, sehingga dari data yang ada, kita mengelompokan 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[6].

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 perinta pip install -U scikit-learn seperti gambar berikut. Lihat pada gambar 1.10.
8. Lalujika 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 estimato/parameter dengan suatu nilai load berfungsi untuk mengulang data sebelumnya)

- dari ketiga baris akhir tersebut jika di jalankan atau 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 package `sklearn`).
- `from sklearn.svm import SVC`
(pada baris ini merupakan sebuah perintah untuk mengimport class `SVC` dari package `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.RandomState(0)).
- `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 OneVsRestClassifier dari packaged sklearn.multiclass).

- from sklearn.preprocessing import LabelBinarizer
(pada baris ini merupakan sebuah perintah untuk mengimport class LabelBinarizer dari package 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 eror

1.4.3.1 ScreenShoot Eror

```
>>> 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 Eror 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

===== jjjjjj HEAD

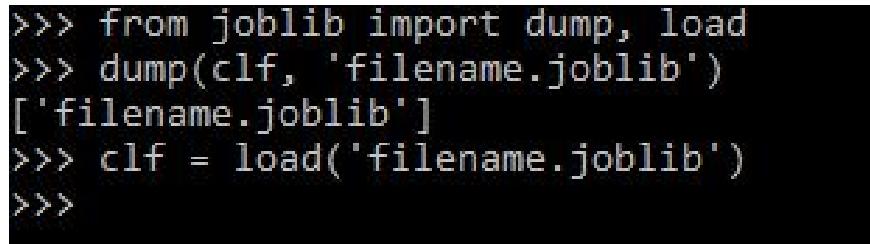


```
Command Prompt
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\HUDA>pip install joblib
Requirement already satisfied: joblib in f:\anaconda\lib\site-packages (0.13.2)
distributed 1.21.8 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.

1.5 Cokro Edi Prawiro/1164069

1.5.1 Praktek teori penunjang

1. Kecerdasan Buatan *Artificial Intelligence* adalah suatu cabang dalam bidang sains komputer yang mengkaji bagaimana untuk melengkapi sebuah komputer dengan kemampuan atau kepintaran seperti manusia. Komputer tersebut diharapkan dapat belajar sendiri dengan cara mengumpulkan data-data yang diterimanya, yang berguna sebagai parameter untuk memecahkan masalah. Jadi kecerdasan buatan merupakan kecerdasan yang di program dalam komputer untuk memecahkan masalah secara tepat dan cepat atau untuk memberikan kemungkinan keberhasilan dan kegagalan pada solusi dari suatu masalah.

Adapun kecerdasan buatan menurut para ahli adalah sebagai berikut :

- Kecerdasan Buatan merupakan Kawasan penelitian, aplikasi dan intruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas (H. A. Simon[1997]).
- Kecerdasan buatan adalah bidang studi yang berhubungan dengan penangkapan, pemodelan, dan penyimpanan kecerdasan manusia dalam sebuah sistem teknologi sehingga sistem tersebut dapat menfasilitasi proses pengambilan keputusan yang biasanya dilakukan oleh manusia (Haag dan keen[1996]).

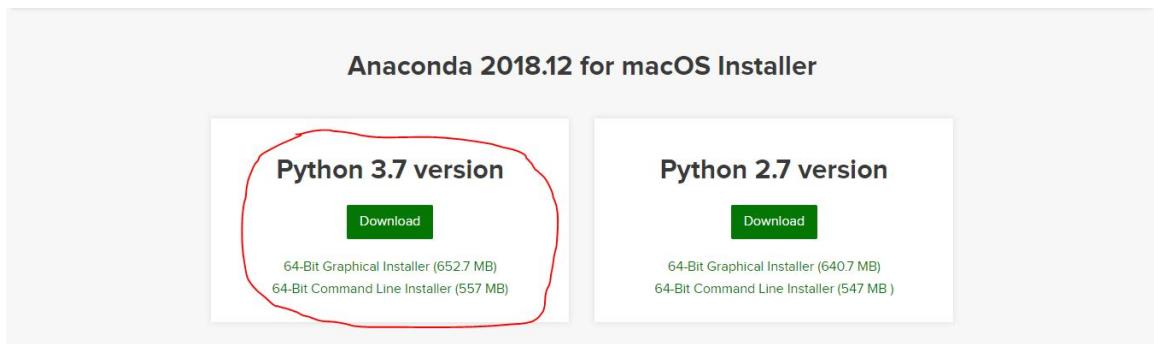


Figure 1.4: Download Anaconda.

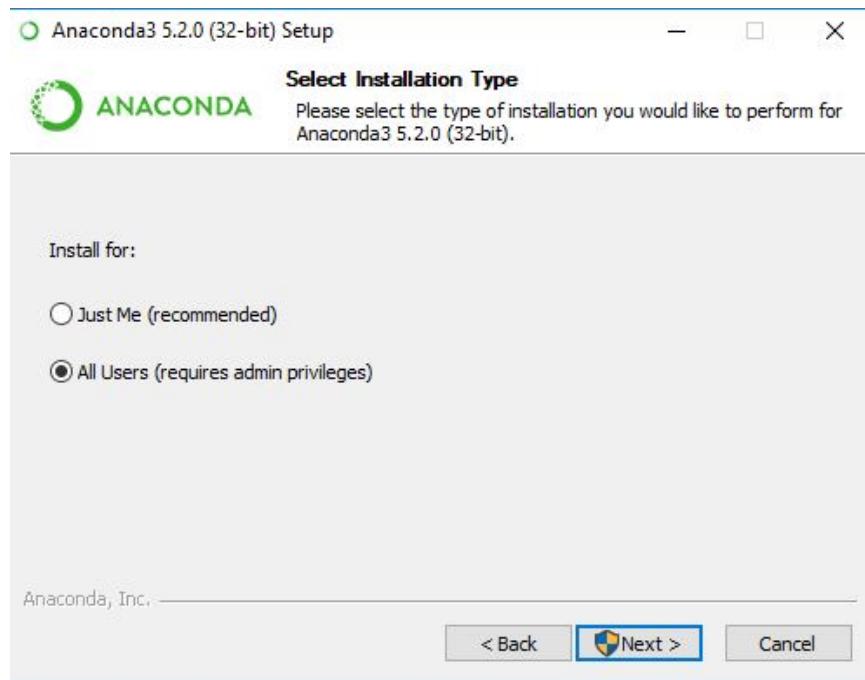


Figure 1.5: Langkah pertama instalasi anaconda.

Sejarah dan Perkembangan Kecerdasan Buatan.

ketika *Rene Descartes* mengemukakan gagasan yang menjadi cikal bakal kecerdasan buatan pada abad 17 mengemukakan bahwa hewan bukan apa-apa melainkan hanya mesin yang rumit yang dilanjutkan oleh *Belaise Pascal* yang telah menciptakan mesin penghitung digital mekanis pertama pada tahun 1642. Lalu pada abad 19 *Charles Babbage* dan *Ada lovelace* bekerja sama membuat mesin penghitung mekanis yang dapat di program.

Perkembangan kecerdasan buatan inipun terus berlanjut, *Bertrand Russell* dan *Alferd North Whithead* menerbitkan *mathematica*, yang merombak logika formal. Setelah itu dilanjutkan dengan penemuan oleh *Warren McCulloch* dan

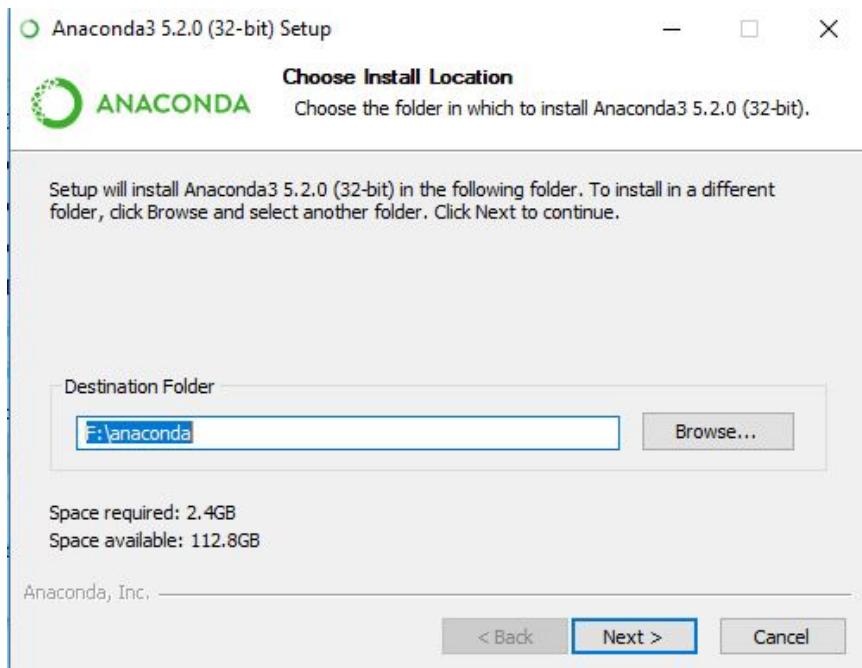


Figure 1.6: Langkah kedua instalasi anaconda.

Walter Pitts menerbitkan Kalkulus Logis Gagasan yang tetap ada dalam Aktivitas pada 1943 yang meletakan pondasi awal berupa jaringan syaraf Kemudian pada tahun 1950-an adalah periode awal usaha aktif kecerdasan buatan. Program Kecerdasan buatan pertama yang bekerja di ciptakan pada tahun 1951 untuk menjalankan mesin Ferranti Mark I di University of Manchester (UK) yaitu sebuah program permainan naskah yang ditulis oleh *Christoper Strachey* dan program permainan catur yang ditulis oleh *Dietric Prinz*. Kemudian pada konferensi pertama tahun 1956 *John McCarthy* mengemukakan istilah kecerdasan buatan kemudian dia juga menemukan bahasa pemrograman lisp. *Joseph Weizenbaum* menciptakan ELIZA, sebuah chatterbot yang menerapkan psikoterapi Rogerian.

Selama rentang waktu tahun 1960-an dan 1970-an, *Joel Moses* mendemonstrasikan kekuatan pertimbangan simbolis untuk mengintegrasikan masalah di dalam program Macsyma, yang merupakan program yang pertamakali sukses dalam bidang matematika. Kemudian pada tahun 1980-an industry kecerdasan buatan ini berkembang walaupun sudah dimulai pada tahun 1970-an Evolusi kecerdasan buatan berjalan dalam dua jalur yang berbeda yaitu meniru proses berpikir manusia untuk menyelesaikan masalah umum. Kedua mengkombinasikan pemikiran terbaik para ahli pada sepotong software yang dirancang

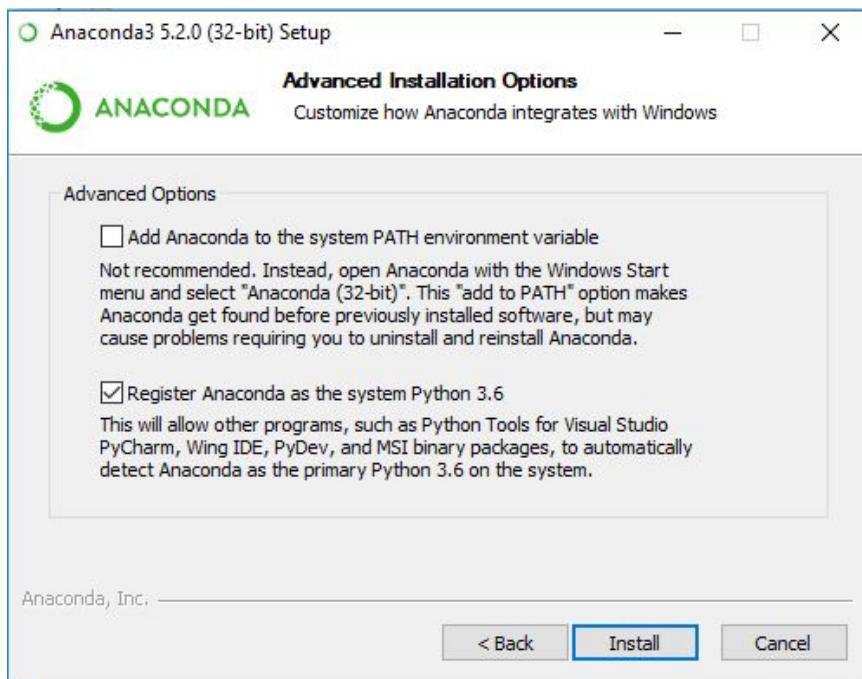


Figure 1.7: Langkah ketiga instalasi anaconda.

untuk memecahkan persolalan yang spesifik.

2. Supervised learning adalah sebuah pendekatan dengan syarat sudah terdapat data yang dilatih kemudian harus terdapat variable yang ditargetkan sehingga tujuan dari pendekatan ini adalah pengelompokan data terhadap data yang telah ada. Ciri khas dari Supervised learning yaitu terdapat label atau nama kelas pada data latih (supervisi) dan data baru di klasifikasikan berdasarkan data latih. Data latih sikelompokan berdasarkan ukuran kemiripan pada suatu kelas. Berdasarkan keluaran dari fungsi, Supervised learning dibagi menjadi 2, regresi dan klasifikasi. Regresi terjadi jika output dari fungsi merupakan nilai yang kontinyu, sedangkan klasifikasi terjadi jika keluaran dari fungsi adalah nilai tertentu dari suatu atribut (tidak kontinyu). Tujuan dari Supervised learning adalah untuk memprediksi nilai dari fungsi untuk sebuah data masukan yang sah setelah melihat sejumlah data latih[3].

Adapun pengertian klasifikasi dan regresi adalah sebagai berikut :

- Klasifikasi merupakan pengelompokan berdasarkan parameter tertentu yang tidak konstan contoh pada mahluk hidup yaitu persamaan ciri cara hidup dan tempat hidup.

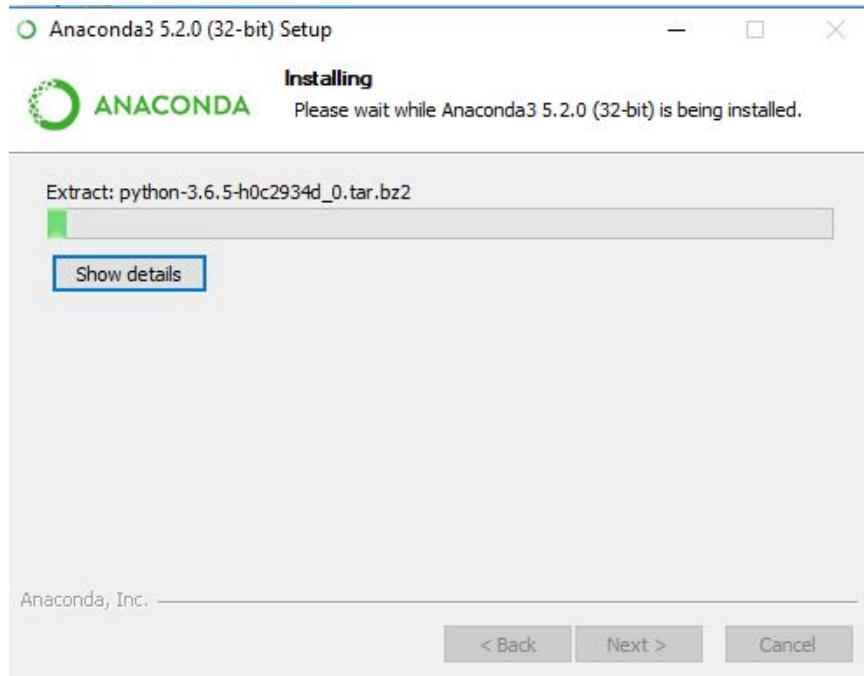


Figure 1.8: Langkah terakhir instalasi anaconda.

```
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.

- Regresi yaitu pengeluaran nilai output yang konstan jika dipicu dengan parameter tertentu biasanya regresi disini berbentuk regresi linier. Regresi linier yaitu metode statistika yang digunakan untuk membentuk model hubungan antara variabel terikat(dependen,respon,Y) dengan satu atau lebih variabel bebas(independent, prdiktor, X). Apabila banyaknya variabel bebas hanya ada satu, disebut sebagai regresi linier sederhana, sedangkan apabila terdapat lebih dari satu variabel bebas, disebut sebagai regresi linier berganda [4].

unsupervised learning adalah pendekatan yang tidak memerlukan data latih atau data training untuk melakukan prediksi atau klasifikasi. Berdasarkan model secara matematisnya, algoritma ini tidak memiliki target variabel. Tu-

```
C:\Users\Huda>pip install -U scikit-learn
Collecting scikit-learn
  Downloading https://files.pythonhosted.org/packages/ee/c8/c89ebcd0d7dbba6e6fd222daabd257da3c28a967dd7c352d4272b2e1cef6/scikit_learn-0.20.2-cp36-cp36m-win32.whl (4.3MB)
    100% |██████████| 4.3MB 425KB/s
Requirement not upgraded as not directly required: numpy>=1.13.2 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
Success: already up-to-date!
You are using pip version 19.0.3, 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.

juan dari algoritma ini yaitu pengelompokan objek yang memiliki kesamaan atau hampir sama dalam satu cakupan wilayah tertentu. Kemudian pada unsupervised learning tidak terdapat label atau nama kelas pada data latih. Kemudian dataset merupakan objek yang menggambarkan data itu sendiri dan relasinya di memory. Struktur datanya mirip dengan struktur data di basis-data. Jadi strikturnya terdiri atas baris kolom dan juga ada sejenis relasi data. Pada dataset terdiri bagian bagian yaitu training set dan Testing set. Adapun pengertian dari training set dan Testing set adalah sebagai berikut :

- Training set adalah bagian dari dataset itu sendiri yang dilatih untuk membuat prediksi atau algoritma mesin learning lainnya sesuai keinginan atau tujuan data itu dibuat.
- Testing set adalah bagian dari dataset yang di tes atau diujicoba untuk melihat keakuratannya dengan katalain melihat peformanya.

```
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.

1.5.2 Instalisasi

Pada proses instalasi ini langkah pertama yaitu mengakses website scikit dengan mengakses link berikut <https://scikit-learn.org/stable/tutorial/basic/tutorial.html> maka hasilnya dapat dilihat pada gambar 1.35 kemudian setelah itu klik button installation maka akan muncul tampilan yang dapat dilihat pada gambar 1.36.

1. cara instalasi Instalasi library scikit dari anaconda langkah pertama instal terlebih dahulu anacondanya dikarenakan anaconda sudah include dengan python maka codingan python dapat digunakan di anaconda dan ketika diperiksa versinya maka akan muncul tampilan seperti Gambar 1.51

kemudian pada cmd administrator install library sikit dengan cara memasukan codingan pip install -U scikit-learn maka hasilnya seperti seperti pada gambar 1.52 berikut.

setelah itu masukan kembali perintah berikut di cmd conda install scikit-learn jika hasilnya seperti pada gambar 1.53 maka librari sikit telah teristal dan siap untuk di gunakan.

```
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.

kemudian untuk mencobanya tuliskan perintah python pada cmd lalu masukan codingan print("Hello Anaconda!") maka hasilnya terlihat seperti gambar 1.54 codingan print("Hello Anaconda!") yaitu berfungsi mencetak nilai yang ada di dalam kurung dan di antara kutip.

2. cara mencoba dataset yaitu dengan cara memasukan perintah berikut pada cmd seperti pada gambar 1.55

- pada codingan from sklearn import datasets menjelaskan import librari dataset dari librari sikit pada gambar 1.55
- pada codingan iris = datasets.load_iris() berarti parameter atau acuan bernama iris kemudian di load ke dalam dataset sebagai perbandingan kalau dalam diagram iris bisa disebut X nya pada gambar 1.55
- pada codingan digits = datasets.load_digits() berarti parameter hampir mirip seperti iris tadi namun digits merupakan kebalikannya kalau di dalam diagaram dia bernilai Y pada gambar 1.55
- kemudian pada codingan print(digits.data) yaitu mencetak data digits yang di bandingkan dengan data iris pada gambar 1.55
- pada codingan print(iris.data) yaitu mencetak data iris yang dibandingkan dengan data digits pada gambar 1.55

3. Mencoba Learning and Predicting

Pada kasus ini dataset digit digunakan untuk memprediksi yang mana telah di berikan gambar untuk mewakili sampel masing-masing dari 10 kelas yang dimulai dari digit nol hingga sembilan yang digunakan untuk memprediksi sampel

```
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 screenshot of a terminal window showing the output of a Python command. The output is the string "array([8])" in white text on a black background.

Figure 1.17: Hasil Tampilan 5.

yang tidak terlihat untuk lebih jelasnya dapat di praktikan codingan berikut ini.

```
>>> from sklearn import datasets
```

pada baris ini dapat diartikan bahwa librari sklearn mengimport package dataset

```
>>> iris = datasets.load_iris()
```

pada baris ini dimasukan parameter iris yang di sandingkan dengan dataset load sehingga iris berisi nilai dataset

```
>>> digits = datasets.load_digits()
```

pada baris ini dimasukan parameter digits yang di sandingkan dengan dataset load sehingga digits berisi nilai dataset

```
>>> from sklearn import svm
```

pada baris ini librari sklearn mengimport package svm

```
>>> clf = svm.SVC(gamma=0.0001, C=100.)
```

pada codingan diatas dibuat variabel clf yang di isi dengan nilai svm dengan nilai gama 0.0001 dan 100.

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

pada codingan clf di implementasikan dengan perintah fit

```
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.



Figure 1.19: Hasil Tampilan 7.

```
SVC(C=100.0, cache_size=200, class_weight=None, coef0=0.0,  
decision_function_shape='ovr', degree=3, gamma=0.0001, kernel='rbf',  
max_iter=-1, probability=False, random_state=None, shrinking=True,  
tol=0.001, verbose=False)
```

yang hasilnya seperti codingan diatas yang menjabarkan isi dari SVC itu sendiri.

```
>>> clf.predict(digits.data[-1:])  
array([8])  
>>>
```

kemudain pada codingan diatas digunakan printah predic yang merupakan printah untuk mengimplementasikan method digits.

4. Mencoba model pertistence

model pertistence yaitu model yang digunakan untuk mengolah data sehingga data tersebut konstan atau konsisten terhadap parameter tertentu contoh pada codingan di bagawah nilai y akan konstan di nol walaupun tetelah di isi nilai lebih dari nol.

```
>>> from sklearn import svm
```

pada baris ini librari sklearn mengimport package svm.

```
>>> from sklearn import datasets
```

pada baris ini librari sklearn mengimport package datasets.



Figure 1.20: Hasil Tampilan 8.

```
>>> 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 1.21: Hasil Tampilan 9.

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

pada codingan diatas dibuat variabel clf yang di isi dengan nilai svm dengan gamma sama dengan scale.

```
>>> iris = datasets.load_iris()
```

pada baris ini dimasukan parameter iris yang di sandingkan dengan dataset load sehingga iris berisi nilai dataset.

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

pada codingan diatas X berisi nilai iris.data dan y berisi nilai iris.target.

```
>>> clf.fit(X, y)
```

method clf di implementasikan dengan perintah fit dengan X, y sebagai nilai untuk implementasinya.

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

Figure 1.22: Hasil Tampilan 10.

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

Figure 1.23: Hasil Tampilan 11.

```
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)
```

maka hasilnya penjabaran dari SVC seperti codingan diatas.

```
>>> import pickle
```

mengimport library atau package pickle.

```
>>> s = pickle.dumps(clf)
```

kemudian di buat variabel s yang di load oleh package pickle dengan di isi nilai clf.

```
>>> clf2 = pickle.loads(s)
```

setelah itu pada codingan diatas dibuat lagi variabel clf2 kemudian di load pickle.

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

kemudian variabel clf2 di implementasikan dengan parameter X dengan nilai 0 berbanding 1 maka hasilnya nilainya array bernilai nol dan y bernilai nol.

```
>>> 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.

```
array([0])
>>> y[0]
0
```

5. mencoba conventions conventions merupakan aturan aturan dasar atau kesepakatan kesepakatan dalam pemrograman sikit python dan anaconda berikut merupakan jenis-jenis codingan conventions :

- Type casting yaitu tipe pelemparan parameter atau variabel kedalam variabel baru.

```
>>> import numpy as np
codingan diatas yaitu import librari numpy yang di inisialisasi menjadi np
>>> from sklearn import random_projection
import librari random_projection
>>> rng = np.random.RandomState(0)
membuat variabel baru dengan nama rng dengan nilai random
>>> X = rng.rand(10, 2000)
memasukan nilai rng kedalam variabel X dengan rad nilai 10 sampai 2000
>>> X = np.array(X, dtype='float32')
menambahkan nilai np berupa array yaitu X dan float 32
>>> X.dtype
dtype('float32')
X.dtype di running menghasilkan nilai dtype float
```

```
>>> 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.

```
>>> transformer = random_projection.GaussianRandomProjection()
membuat variabel transformer dengan nilai random

>>> X_new = transformer.fit_transform(X)
membuat variabel X_new dan di isi nilai transformer kemudian di implementasikan

>>> X_new.dtype
merunning variabel X_new

dtype('float64')
hasil running X_new

>>> from sklearn import datasets
mengimport library dataset

>>> from sklearn.svm import SVC
mengimport library SVC

>>> iris = datasets.load_iris()
membuat variabel iris dengan nilai load dataset

>>> clf = SVC(gamma='scale')
membuat variabel clf bernilai SVC dengan gamma menggunakan nilai sekala

>>> clf.fit(iris.data, iris.target)
```

```
>>> 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.

merunning variabel atau method clf dengan isian nilai iris data dan 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)
```

detail hasil runing clf

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

memunculkan detail atau lis sebanyak tiga nilai

```
>>> clf.fit(iris.data, iris.target_names[iris.target])
```

merunning kembali clf dengan nilai iris data, iris target name

```
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)
```

hasil dari running clf

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

memunculkan tiga nilai yang telah dilempar dari SVC

```
>>> 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.

- Refitting and updating parameters atau pengisian ulang atau memperbaiki paramater merupakan cara untuk merubah nilai dari sebuah parameter contoh nilai x adalah 10 jika di perbarui bisa menjadi 15 begitu juga dalam codiangan berikut hal ini dapat dilakukan untuk lebih jelasnya dabat dilihat codingan dibawah ini :

```
>>> import numpy as np
codingan diatas yaitu import librari numpy yang di inisialisasi menjadi np
>>> from sklearn.svm import SVC
mengimport library SVC
>>> rng = np.random.RandomState(0).
membuat variabel baru dengan nama rng dengan nilai random.
>>> X = rng.rand(100, 10)
paramater X dengan nilai dari variabel rng dan rad dari 100 sampai 10.
>>> y = rng.binomial(1, 0.5, 100)
parameter y dengan nilai rng binominal dari 1 0,5 sampai 100.
>>> X_test = rng.rand(5, 10)
parameter X_test dengan nilai rng dan rad dari 5 ke 10
>>> clf = SVC()
parameter clf bernilai SVC
>>> clf.set_params(kernel='linear').fit(X, y)
```

```
>>> 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.

parameter clf di set dengan mengkompile atau mengekstrak nilai X dan y dengan kernel linear.

```
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)
```

penjabaran nilai SVC hasil running CLF

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

meranning clf dengan nilai X_test

```
>>> clf.set_params(kernel='rbf', gamma='scale').fit(X, y)
```

parameter clf di set dengan kernel rbf dan gama skala dan mengkompile nilai X dan 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)
```

penjabaran nilai SVC hasil running CLF

```
>>> 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.

The screenshot shows the homepage of the scikit-learn website at <https://scikit-learn.org/stable/tutorial/basic/tutorial.html>. The page title is "An introduction to machine learning with scikit-learn". The left sidebar contains navigation links for "Previous scikit-learn", "Next scikit-learn", "Up scikit-learn", "scikit-learn v0.20.2 Other versions", and a "Please cite us if you use the software." button. The main content area starts with a section titled "Machine learning: the problem setting". It explains that a learning problem considers a set of `n` samples of data and tries to predict properties of unknown data. It distinguishes between supervised learning (with additional attributes) and regression (continuous output). It also mentions classification (discrete categories) and multiclass vs. multilabel fitting.

Figure 1.35: Tampilan website Scikit 1.

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

Hasil dari running `clf`.

- Multiclass vs. multilabel fitting perbandingan antara banyak klass dan pelabelan yang tepat berikut merupakan codingannya.

```
>>> from sklearn.svm import SVC
```

mengimport library SVC

```
>>> from sklearn.multiclass import OneVsRestClassifier
```

memasukan librari OneVsRestClassifier dengan kondisi multi class.

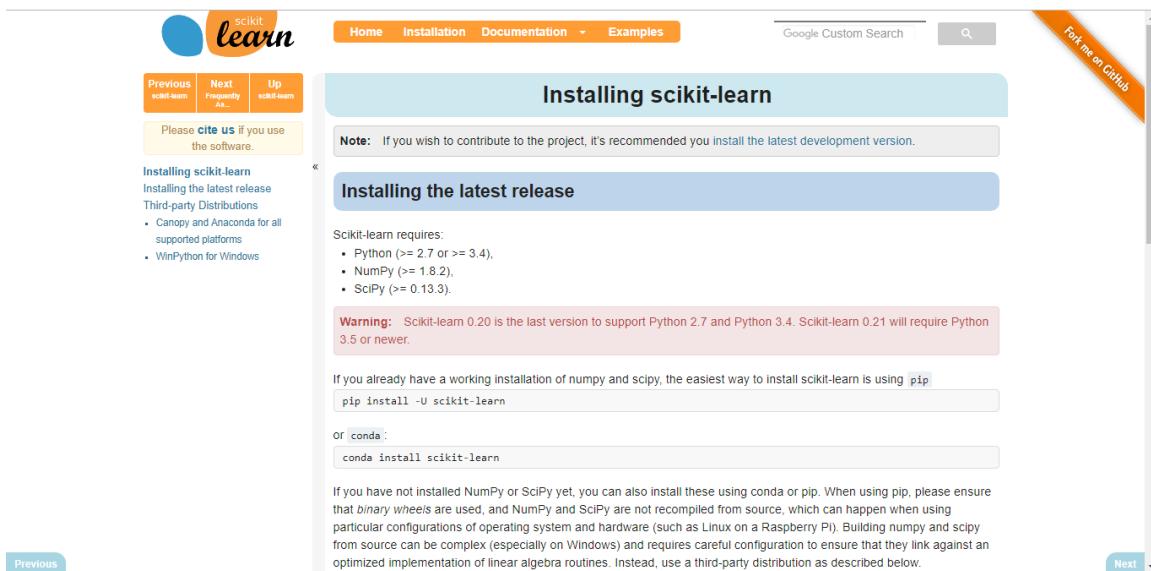


Figure 1.36: Tampilan website Scikit 2.

```
>>> from sklearn.preprocessing import LabelBinarizer
memasukan librari LabelBinarizer

>>> X = [[1, 2], [2, 4], [4, 5], [3, 2], [3, 1]]
pemberian nilai pada parameter X

>>> y = [0, 0, 1, 1, 2]
pemberian nilai pada parameter y

>>> classif = OneVsRestClassifier(estimator=SVC(gamma='scale',
... random_state=0))

opsi untuk class dengan ketentuan estimator SVC gamma berbentuk skala
dan random (acak).

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

>>> y = LabelBinarizer().fit_transform(y)
memberikan nilai pada parameter y

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

```
[1, 0, 0],  
[0, 1, 0],  
[0, 0, 1],  
[0, 0, 0]])
```

hasil running classif

```
>>> from sklearn.preprocessing import MultiLabelBinarizer
```

import librari multi label

```
>>> y = [[0, 1], [0, 2], [1, 3], [0, 2, 3], [2, 4]]
```

memberikan nilai pada parameter y

```
>>> y = MultiLabelBinarizer().fit_transform(y)
```

membuat parameter y menjadi multi label.

```
>>> classif.fit(X, y).predict(X)
```

1.6 Fathi Rabbani / 1164074

1.6.1 Teori

1. Sejarah dan Perkembangan Kecerdasan Buatan

Sejarah dari sebuah Artificial Intelligence atau dalam Bahasa indonesianya diterjemahkan sebagai Kecerdasan Buatan adalah sebuah usaha untuk dapat memodelkan sebuah mesin agar dapat berfikir dan menirukan tingkah laku dan cara berfikir manusia, ada beberapa jenis dari kecerdasan buatan, yaitu :

- Symbol Manipulating AI
- Nueral AI
- Neural Network

Peneliti yang selalu disebutkan sebagai Bapak AI adalah Jhon McCharty merupakan seorang dosen yang mengenalkan Kecerdasan Buatan kepada 2 lembaga penelitian hebat, yaitu Stanford Artificial Intelligence Laboratory dan MIT Artificial Intelligence Laboratory.

Sedangkan perkembangan kecerdasan buatan saat ini sudah mencapai tahap dimana manusia mulai membuat sebuah robot yang dapat menirukan hampir

90 persen dari keseharian mereka, mulai dari bidang kesehatan, koki, pabrik, kantor, hingga sebuah robot yang bertugas sebagai seorang pelayan di sebuah restoran. Dan dubai sebagai pengguna mobil tanpa pengemudi yang menerapkan AI dengan menggunakan data wilayah serta jarak kendaraan dengan pingir jalan.

2. Definisi Supervised, Unsupervised Learning, Klasifikasi, Regresi serta Data, Training, Testing Set

- Supervised learning merupakan sebuah pendekatan AI dengan latihan yang sudah dilakukan dengan sebuah data yang lengkap, dan memiliki variable yang dapat digunakan sebagai target sehingga dapat menujukan data agar menjadi kelompok dari sebuah data menjadi kelompok data yang baru.
- Unsupervised learning merupakan sebuah pendekatan AI tanpa menggunakan data yang lengkap dan ter-variable sehingga harus dilakukan pengelompokan agar data tersebut dapat digunakan.
- Klasifikasi merupakan sebuah pengelompokan suatu objek ke dalam kategori tertentu.
- Regresi merupakan pendekatan model matematika untuk mendeskripsikan hubungan dari beberapa variabel independen dengan variable dependen.
- Data Set, merupakan sebuah objek yang merepresentasikan data dan hubungannya di memory.
- Training Set, subset untuk melatih model.
- Testing Set, subset untuk menguji model yang sudah dilatih.

1.6.2 Praktikum

3. Instalasi Library Scikit dari Anaconda

Pertama Download terlebih dahulu anaconda-nya di <https://www.anaconda.com/distribution/> pilih Operating Sistem yang kalian gunakan. lalu setelah download Install dengan proses berikut :

- Proses Instalasi Anaconda pada gambar 2.73 hingga proses 2.80.
- Proses Instalasi Scikit-Learn dengan menggunakan Conda pada gambar 2.81 hingga gambar 2.83.

- contoh dari Variable Explorer yang digunakan ada pada gambar 1.48.

4. Load Example Dataset dan Menjeleaskan kegunaan barisan Code

berikut ini adalah contoh dataset yang digunakan untuk melakukan compile ada pada gambar 1.49 dan hasilnya ada pada gambar 1.50.

- dari code yang dicoba diketahui bahwa data set yang digunakan adalah data yang diambil dari SKLEARN yang ada pada gambar 1.49.
- Learning and Predicting

Dalam scikit-learn estimator untuk klasifikasi adalah sebuah objek data python yang memiliki implementasi method fit(x, y) dan predict(T). Sebuah estimator dari class sklearn.svm.SVC, yang mana implemetasi dari support vector classification. Estimator dari konstruktor mengambil argument dari model parameter.

```
from sklearn import svm
clf = sv.SVC(gamma=0.001, C=100.)
clf.fit(digits.data[:-1], digits.target[:-1]
clf.predict(digits.data[:-1])
```

mengambil nilai data svm ada pada class sklearn, lalu set nilai data dengan clf = sv.SVC(gamma=0.001, C=100.). variable clf digunakan dengan method fit yang di set nilainya [-1] yang memproduksi array baru dari data digits.data, dengan menggunakan digits.data sebagai acuan, sekarang tinggal melakukan prediksinya.

- Model Persistence

```
from sklearn import svm
from sklearn import datasets
clf = svm.SVC(gamma=0.001)
iris = datasets.load_iris()
X, y = iris.data, iris.target
clf.fit(X, y)
```

mengambil nilai data svm ada pada class sklearn dan mengambil nilai data datasets ada pada class sklearn, lalu buat variable clf dengan nilai data dan buat variable yang berisi nilai load_iris. variable X dan y yang

berisi nilai iris data dan iris target, lalu memanggil nilai variable X dan y dengan data variable clf dan method fit.

```
import pickle
s = pickle.dumps(clf)
clf2 = pickle.loads(s)
clf2.predict(X[0:1])
y[0]
```

mengambil nilai data pickle dan membuat variable s dengan data nilai pickle.dumps yang berisi data variable clf, membuat variable clf2 dengan data pickle.loads yang menggunakan variable s. menggunakan data variable clf2 dengan method predict dengan data variable X dan data variable y.

- Conventions

- ```
import numpy as np
from sklearn import random_projection

rng = np.random.RandomState(0)
X = rng.rand(10, 2000)
X = np.array(X, dtype='float32')
X.dtype

transformer = random_projection.GaussianRandomProjection()
X_new = transformer.fit_transform(X)
X_new.dtype
```

mengambil data numpy dan dialiaskan sebagai np. dari data sklearn mengambil data random\_projection. lalu buat variable rng yang berisi nilai data np dengan random yang berawal 0. lalu variable X dengan data rng yang memiliki type rand berisi data 10 dan 2000. lalu dibuatkan arraynya dengan format X = np.array(X, dtype = 'float32'). dan nilai variable transform yang digunakan untuk menampilkan hasil random, dengan format Gaussian random projection. format penampilannya X\_new = transformer.fit\_transform( X ) dan X\_new.dtype

- ```
from sklearn import datasets
from sklearn.svm import SVC
```

```
iris = datasets.load_iris()
clf = SVC(gamma=0.001)
clf.fit(iris.data, iris.target)

list(clf.predict(iris.data[:3]))

clf.fit(iris.data, iris.target_names[iris.target])

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

dari data sklearn mengambil datasets dan SVC dari svm, selanjutnya membuat format data variable dengan nilai load_iris, clf dengan format SVC $\gamma = 0.001$, lalu di jalankan dengan method clf.fit dengan iris.data dan iris.target sebagai nilainya.

buatkan tampilan datanya dengan list menampilkan data clf dengan predict pada iris.data, dan dilakukan selanjutnya dengan clf.fit dengan nilai data iris.data dan iris.target_names[iris.target]. tampilkan lagi dalam bentuk list data tersebut.

```
- import numpy as np
from sklearn.svm import SVC

rng = np.random.RandomState(0)
X = rng.rand(100, 10)
y = rng.binomial(1, 0.5, 100)
X_test = rng.rand(5, 10)

clf = SVC()
clf.set_params(kernel='linear').fit(X, y)

clf.predict(X_test)
```

```
clf.set_params(kernel='rbf', gamma=0.001).fit(X, y)
```

```
clf.predict(X_test)
```

mengambil data numpy dan dialiaskan sebagai np dan dari sklearn.svm mengambil data SVC, lalu buat variable rng yang berisi nilai data np

dengan random yang berawal 0, lalu variable X dengan data rng yang memiliki type rand berisi data(100, 10) dan y yang memiliki data (1, 0.5, 100) dengan type data binomial lalu buat X_test untuk variable test.

```
- from sklearn.svm import SVC
from sklearn.multiclass import OneVsRestClassifier
from sklearn.preprocessing import LabelBinarizer

X = [[1, 2], [2, 4], [4, 5], [3, 2], [3, 1]]
y = [0, 0, 1, 1, 2]

classif = OneVsRestClassifier(estimator=SVC(gamma=1, random_state=0))

classif.fit(X, y).predict(X)

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

mengambil data sklearn SVC dari svm, OneVsRestClassifier dari multiclass, dan LabelBinarizer dari preprocessing. lalu buatkan variable X dan y yang berisi data nilai dan buatkan data variable clasif untuk digunakan sebagai parameter bagi OneVsRestClassifier yang menghitung data estimator SVC. lalu jalankan dengan method fit dan predict, lalu variable y digunakan sebagai parameter yang menjalankan method fit_transform yang berisi data LabelBinarizer.

```
- 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)
```

mengambil data dari sklearn datanya adalah MultiLabelBinarizer dari preprocessing, buatkan variable y ayng berisikan nilai integer untuk di proses agar menghasilkan data seperti berikut

```
>>>>> c6dbbab89a86daa2bf691a75d9a15c4b56fb15d7
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]])  
||||| HEAD hasil running classif setelah nilai parameter y telah di ganti.
```

1.6.3 Penanganan Error

- (a) Screenshot Error untuk lebih jelasnya Screenshot codingan dapat dilihat pada gambar 1.56 dan 1.57
- (b) Kode error pada screenshot untuk kode error pada gambar 1.56 yaitu pada source code berikut

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

pada codingan tersebut menjadi error dikarenakan variabel atau method digits belum di definisikan. sedangkan untuk kode error pada gambar 1.57 yaitu pada source code

```
from joblib import dump, load
```

pada codingan tersebut terjadi error dikarenakan module joblib belum di install atau modul tersebut tidak ada di library python.

- (c) Solusi Pemecahan Masalah Error

- untuk memperbaiki error pada gambar 1.56 tinggal mendefinisikan variabel atau method digits, untuk lebih lengkapnya dapat dilihat pada gambar 1.58 dengan cara tersebut maka masalah error dapat diselesaikan.
- sedangkan untuk error joblib bisa dilakukan dengan cara masuk ke cmd administrator kemudian isikan perintah pip install joblib kemudian tekan enter sehingga hasilnya terlihat seperti gambar 1.59 setelah itu coba masuk kembali ke python di cmd dan ketikan perintah from joblib import dump, load maka hasilnya seperti gambar 1.60.

=====

5. Error Handling

- (a) Screenshot
- (b) Error Code and Error Type
 - Module Not Found
 - module yang dicari tidak ditemukan, karena file yang dicari tidak ada atau belum di instal.

- Type Data Error

type data yang seharusnya diisikan oleh data number namun diisikan oleh data str/character sedangkan nilai yang bisa dibaca adalah number.

(c) Solution

- For Module Not Found

lakukan instalasi dengan memasukan code berikut untuk download dan instalasi module JOBLIB

```
conda install -c anaconda joblib
```

- For Data Type Error

ganti isi data menjadi data number, contoh :

```
clf = SVC(gamma='scale')
```

ganti menjadi

```
clf = SVC(gamma=0.5)
```

lliili c6dbbab89a86daa2bf691a75d9a15c4b56fb15d7 lliili b1d7802739e393f94f14f567f8e8c88ea

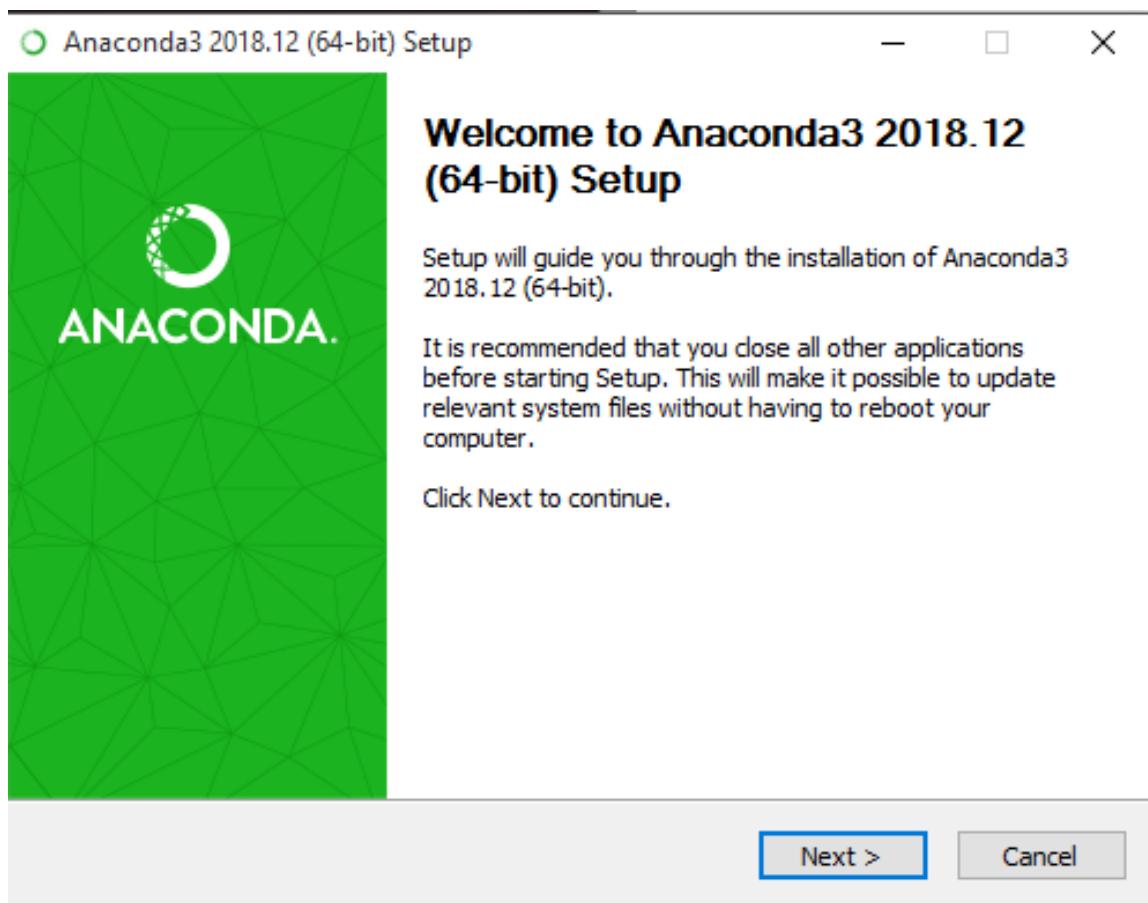


Figure 1.37: setelah membuka data instalasi klik next

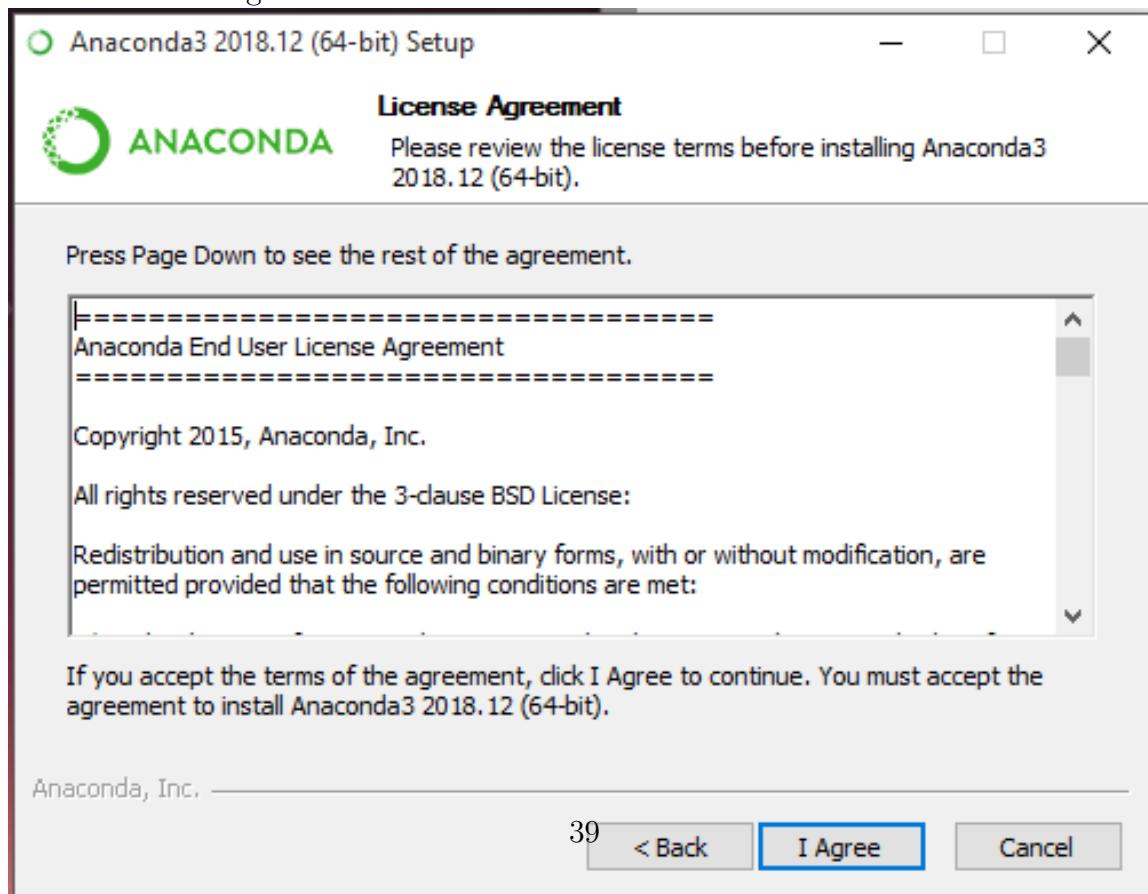


Figure 1.38: pilih i agree

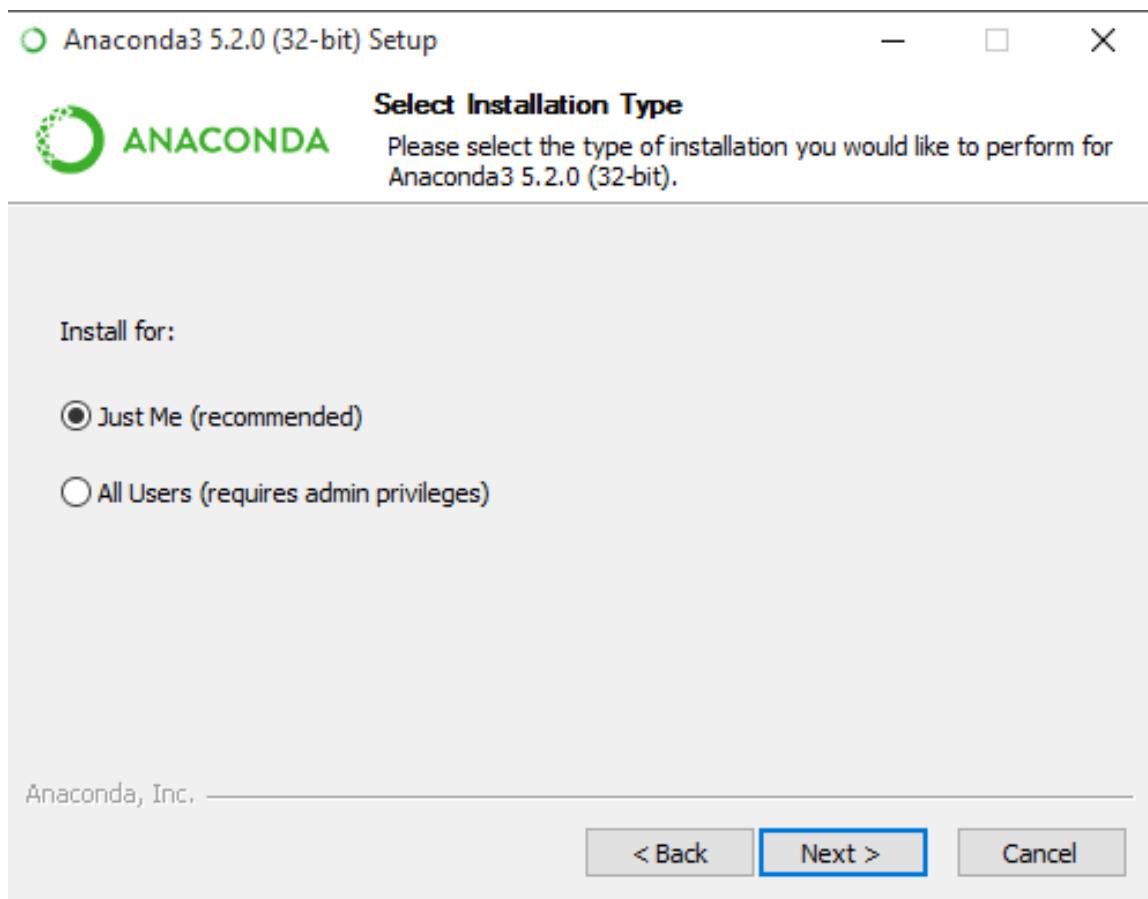


Figure 1.39: pilih instalasi Just Me

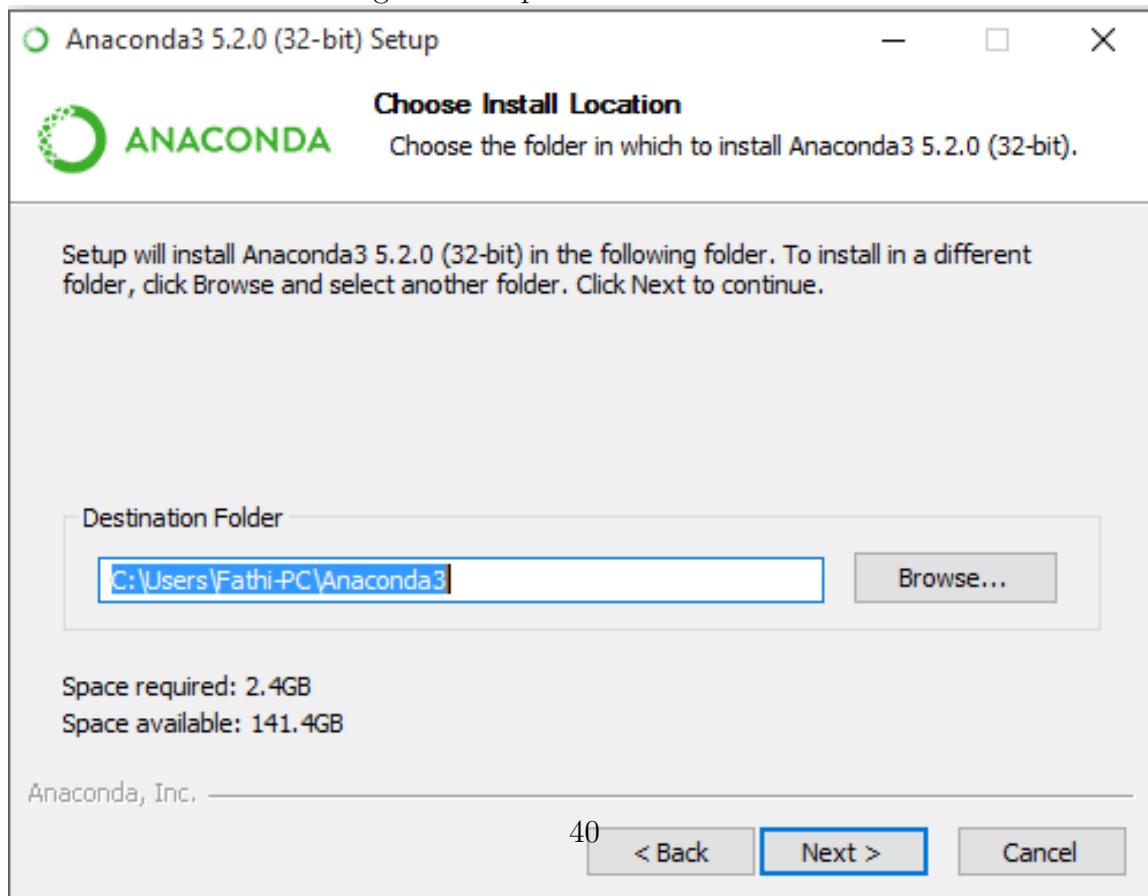


Figure 1.40: langsung saja next

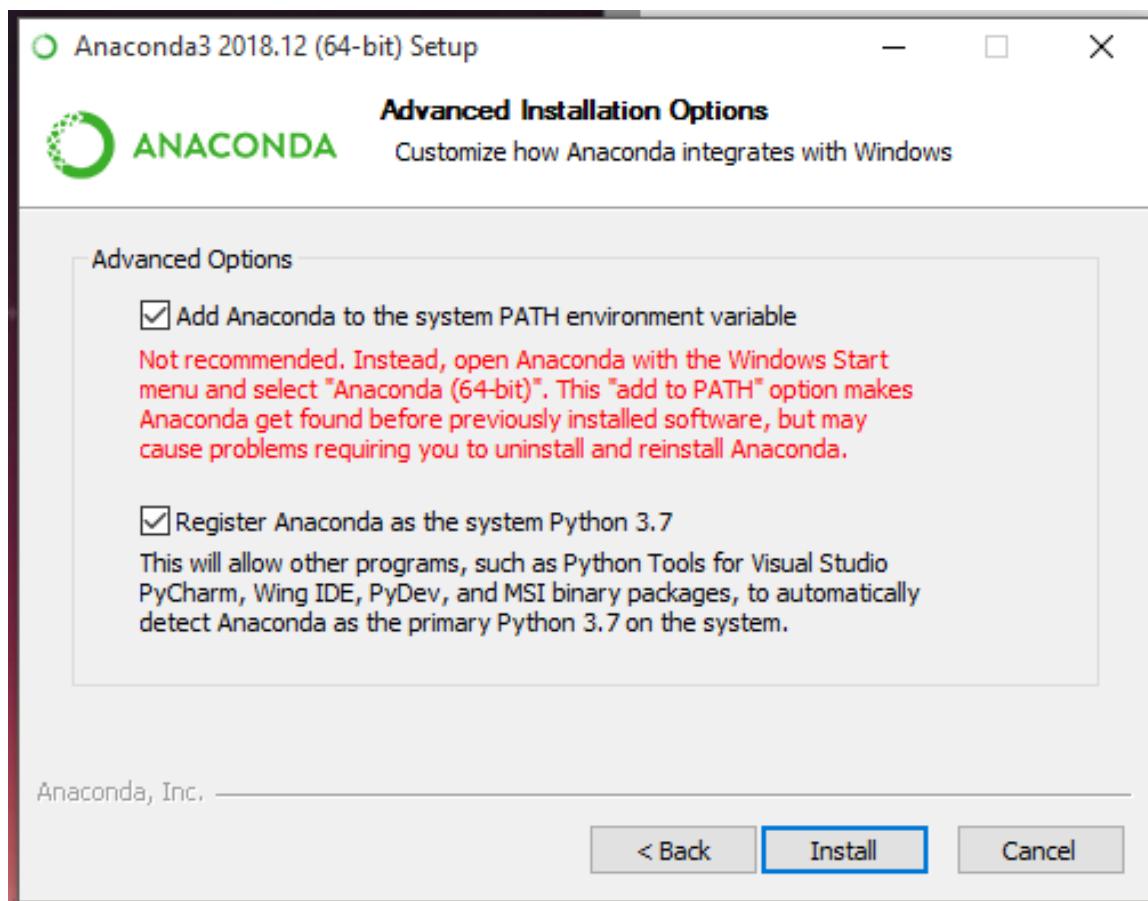


Figure 1.41: cek kedua pilihan tersebut

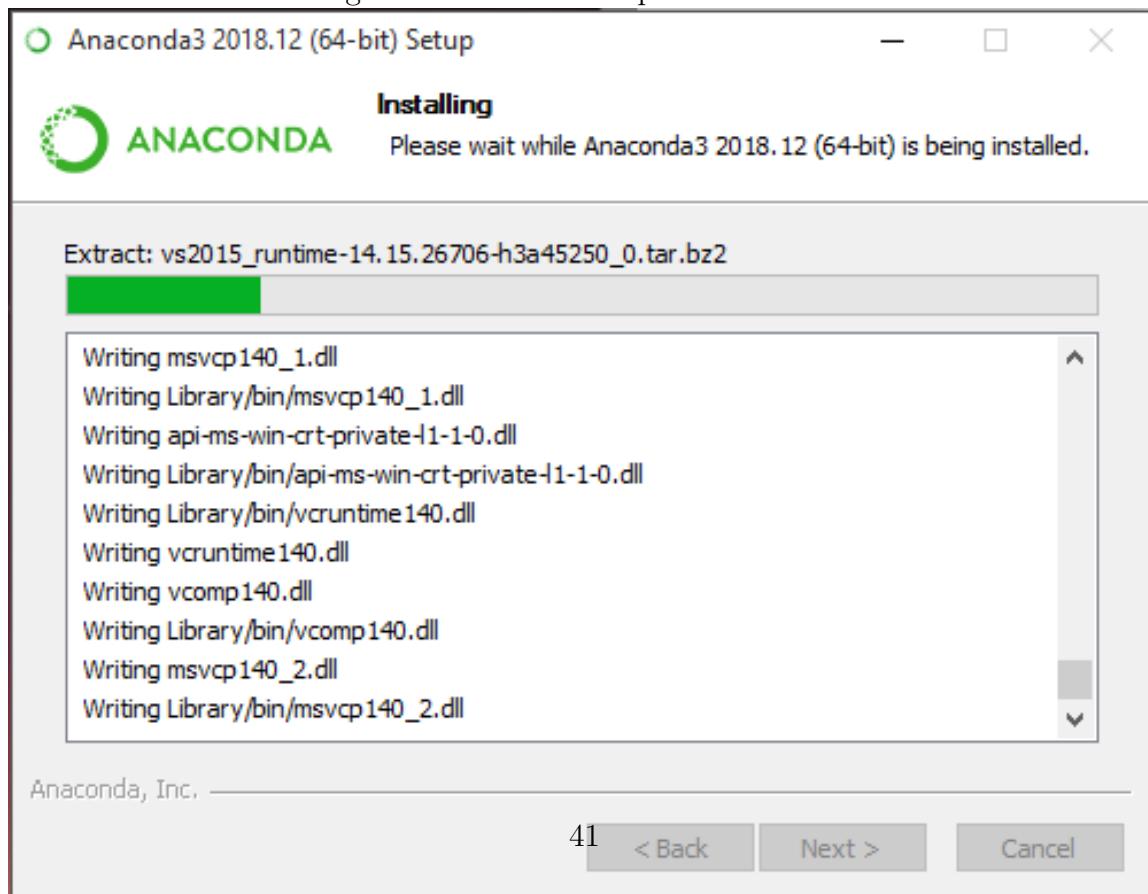


Figure 1.42: proses Instalasi

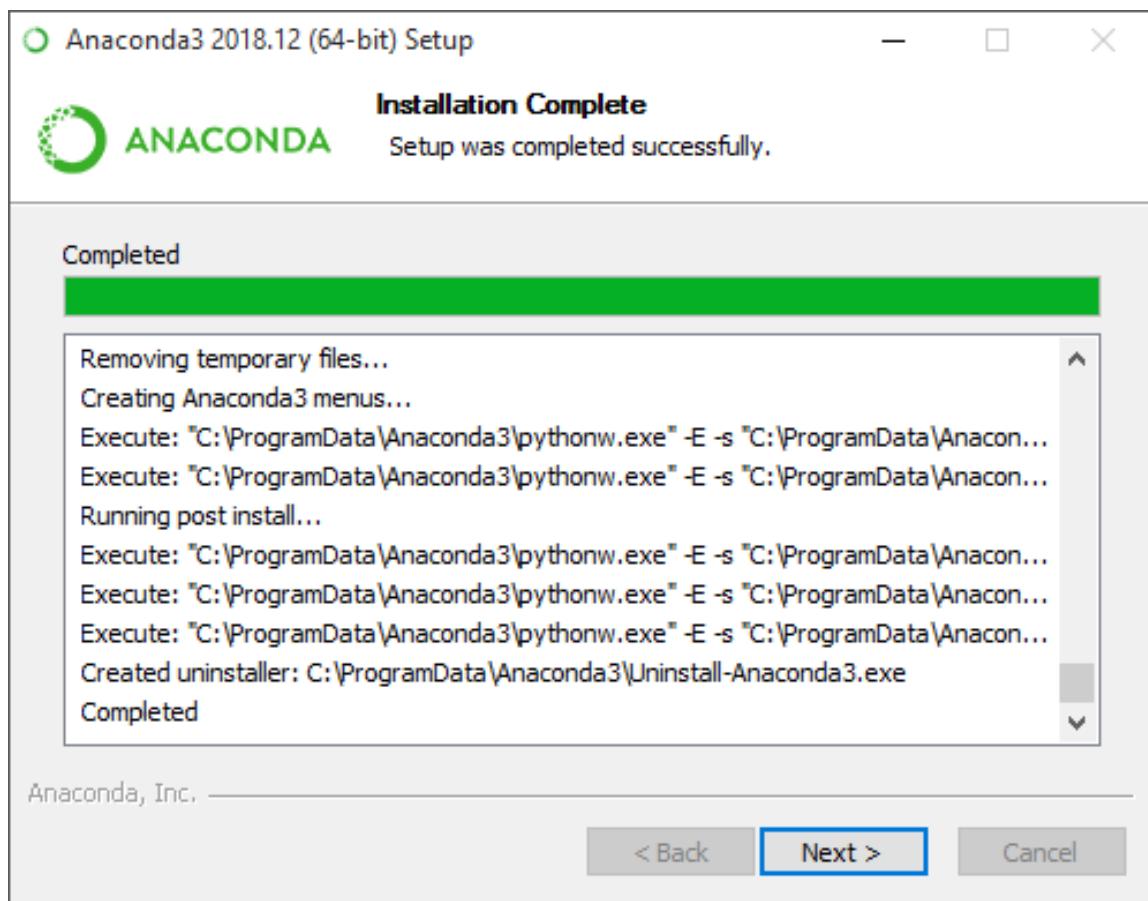


Figure 1.43: klik next

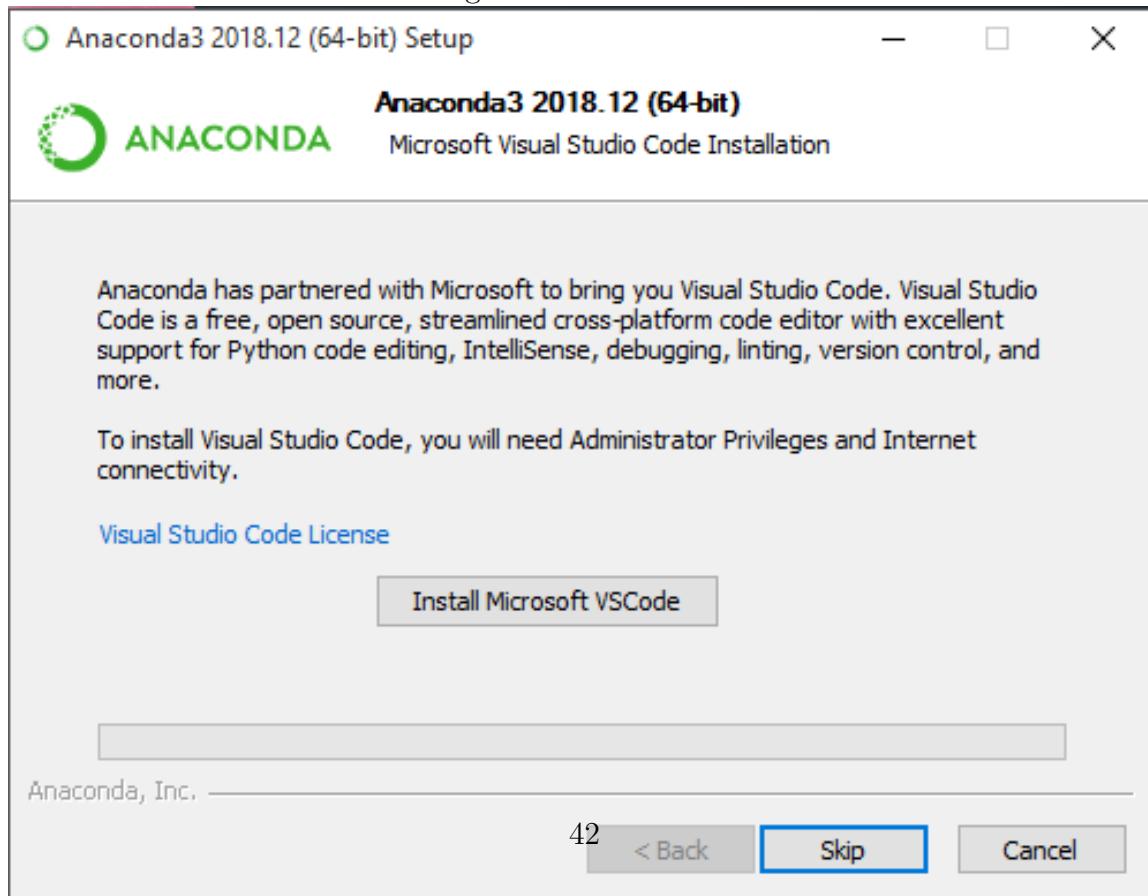


Figure 1.44: selesai instalasi anaconda

```
C:\Users\Fathi-PC>conda --version  
conda 4.5.4  
  
C:\Users\Fathi-PC>python --version  
Python 3.6.5 :: Anaconda, Inc.  
  
C:\Users\Fathi-PC>conda install scikit-learn  
Solving environment: done
```

Figure 1.45: Instalasi SCIKIT dengan menggunakan anaconda

```
Solving environment: done  
  
## Package Plan ##  
  
environment location: C:\Users\Fathi-PC\Anaconda3  
  
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
```

Figure 1.46: Konfirmasi Instalasi

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

Figure 1.47: hasil dari instalasi SCIKIT

| Name | Type | Size | Value |
|--------|-------------|------|--------------------------------------|
| digits | utils.Bunch | 5 | Bunch object of sklearn.utils module |
| iris | utils.Bunch | 5 | Bunch object of sklearn.utils module |

Figure 1.48: data variable explorer

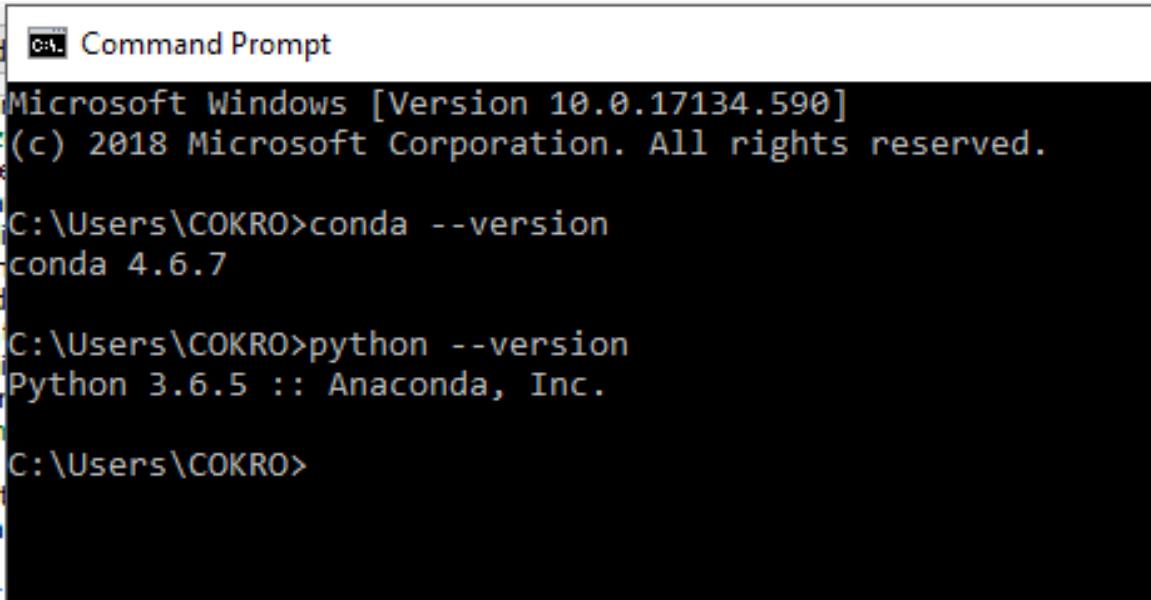
```
from sklearn import datasets
iris = datasets.load_iris()
digits = datasets.load_digits()

print(digits.data)
```

Figure 1.49: code example dataset yang digunakan

```
In [14]: runfile('C:/Users/Fathi-PC/.spyder-py3/temp.py', wdir='C:/Users/Fathi-PC/.spyder-py3')
[[ 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 1.50: data hasil dari code example dataset yang digunakan



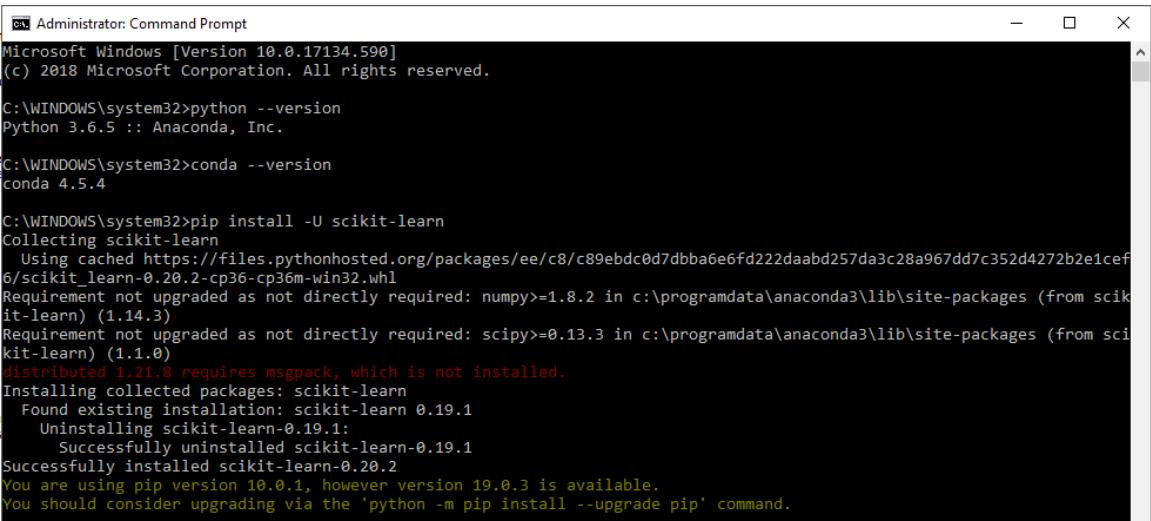
```
Command Prompt
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\COKRO>conda --version
conda 4.6.7

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

C:\Users\COKRO>
```

Figure 1.51: Tampilan Versi Python dan Anaconda .



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

C:\WINDOWS\system32>python --version
Python 3.6.5 :: Anaconda, Inc.

C:\WINDOWS\system32>conda --version
conda 4.5.4

C:\WINDOWS\system32>pip install -U scikit-learn
Collecting scikit-learn
  Using cached https://files.pythonhosted.org/packages/ee/c8/c89ebdc0d7dbba6e6fd222daabd257da3c28a967dd7c352d4272b2e1cef6/scikit_learn-0.20.2-cp36-cp36m-win32.whl
Requirement not upgraded as not directly required: numpy>=1.8.2 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (1.14.3)
Requirement not upgraded as not directly required: scipy>=0.13.3 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (1.1.0)
distributed 1.21.8 requires msgpack, which is not installed.
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 1.52: Instalisasi Library Sikic.

```
C:\WINDOWS\system32>conda install scikit-learn
Solving environment: done

## Package Plan ##

  environment location: C:\ProgramData\Anaconda3

  added / updated specs:
    - scikit-learn

The following packages will be UPDATED:

  conda: 4.5.4-py36_0 --> 4.6.7-py36_0

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

C:\WINDOWS\system32>
```

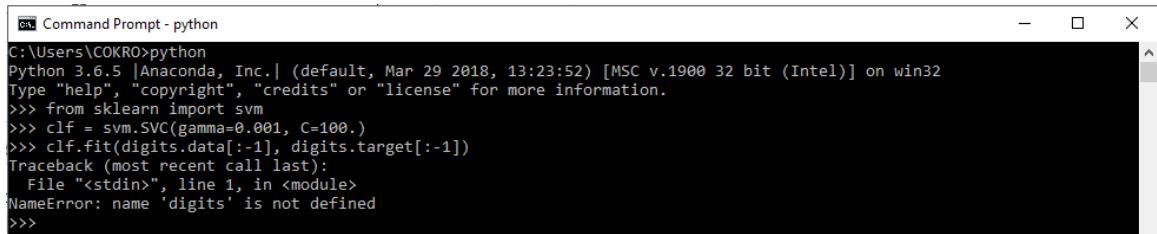
Figure 1.53: Instalasi Library Sikic Melalui Conda

```
C:\Users\COKRO>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.
>>> print("Hello Anaconda!")
Hello Anaconda!
>>>
```

Figure 1.54: Console Python Include Anaconda

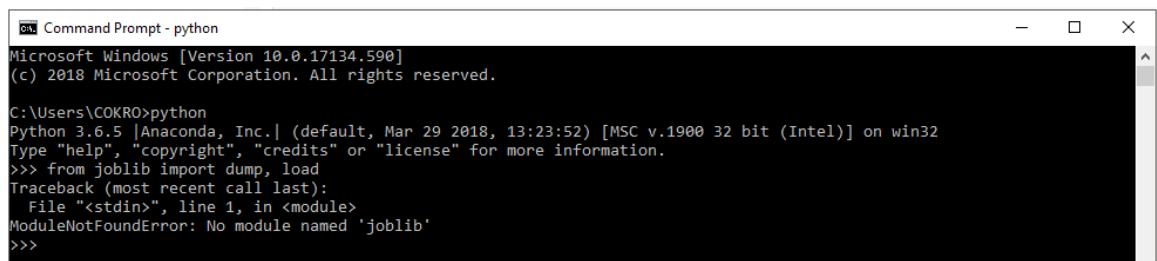
```
C:\ Command Prompt - python
C:\Users\COKRO>python
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:23:52) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> digits = datasets.load_digits()
>>> print(digits.data)
[[ 0.  0.  5. ...  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.]]
>>> print(iris.data)
[[5.1 3.5 1.4 0.2]
 [4.9 3. 1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5. 3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5. 3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3. 1.4 0.1]
 [4.3 3. 1.1 0.1]
 [5.8 4. 1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1. 0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5. 3. 1.6 0.2]
 [5. 3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]]
```

Figure 1.55: Contoh Codingan Dataset



```
C:\Users\COKRO>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 import svm
>>> clf = svm.SVC(gamma=0.001, C=100.)
>>> clf.fit(digits.data[:-1], digits.target[:-1])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'digits' is not defined
>>>
```

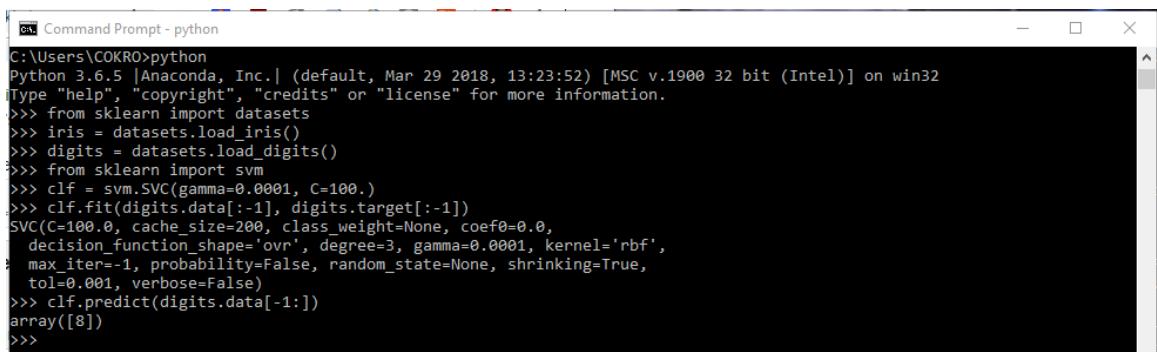
Figure 1.56: Error Coding 1



```
C:\Users\COKRO>python
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

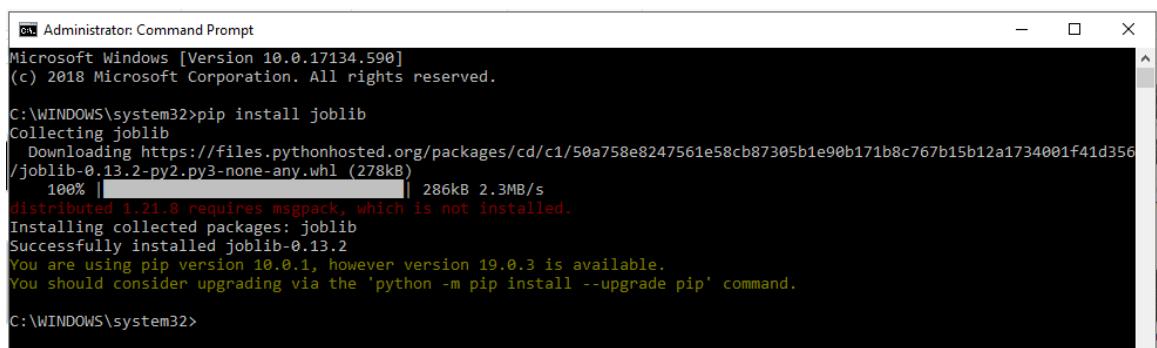
C:\Users\COKRO>python
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:23:52) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from joblib import dump, load
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'joblib'
>>>
```

Figure 1.57: Error Coding 2



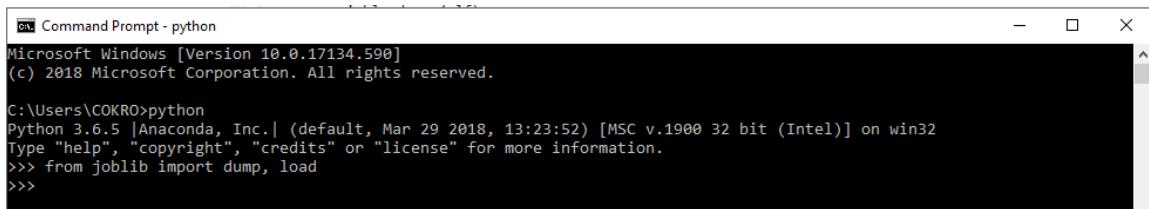
```
C:\Users\COKRO>python
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:23:52) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> digits = datasets.load_digits()
>>> from sklearn import svm
>>> clf = svm.SVC(gamma=0.0001, C=100.)
>>> clf.fit(digits.data[:-1], digits.target[:-1])
SVC(C=100.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.0001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
>>> clf.predict(digits.data[-1:])
array([8])
>>>
```

Figure 1.58: Codingan Solusi Untuk Error digits



```
C:\WINDOWS\system32>pip install joblib
Collecting joblib
  Downloading https://files.pythonhosted.org/packages/cd/c1/50a758e8247561e58cb87305b1e90b171b8c767b15b12a1734001f41d356/joblib-0.13.2-py2.py3-none-any.whl (278kB)
    100% |██████████| 286kB 2.3MB/s
distributed 1.21.8 requires msgpack, which is not installed.
Installing collected packages: joblib
Successfully installed joblib-0.13.2
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.
C:\WINDOWS\system32>
```

Figure 1.59: Codingan Solusi Untuk Error Joblib



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

C:\Users\COKRO>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 joblib import dump, load
>>>
```

Figure 1.60: Hasil Solusi Error Joblib

```
>>> from sklearn import svm
>>> from sklearn import datasets
>>> clf = svm.SVC(gamma='scale')
>>> iris = datasets.load_iris()
>>> X, y = iris.data, iris.target
>>> clf.fit(X, y)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "C:\Users\Fathi-PC\Anaconda3\lib\site-packages\sklearn\svm\base.py", line 187,
in fit
    fit(X, y, sample_weight, solver_type, kernel, random_seed=seed)
  File "C:\Users\Fathi-PC\Anaconda3\lib\site-packages\sklearn\svm\base.py", line 254,
in _dense_fit
    max_iter=self.max_iter, random_state=random_state)
  File "sklearn\svm\libsvm.pyx", line 58, in sklearn.svm.libsvm.fit
TypeError: must be real number, not str
```

Figure 1.61: Error Type data, yang harus digunakan number sedangkan isinya 'SCALE' pada gamma

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

Figure 1.62: Error no Module found, modul yang dicari tidak ditemukan atau tidak ada 'JOBLIB'

Chapter 2

Related Works

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

2.1 Cokro Edi Prawiro/ 1164069

2.1.1 Teori

1. Jelaskan Apa Itu binari classification drlengkapi ilustrasi gambar sendiri.

Binary Classification atau biominal adalah tugas mengklasifikasikan unsur unsur dari himpunan yang diberikan kedalam kedua kelompok berdasarkan aturan klasifikasi yang telah ditetapkan. binari clasification juga dapat diartikan sebagai pembagi yang hanya memberikan dua pilihan contohnya benar dan salah atau klasifikasi tongkat panjang atau pendek. penjelasan lebih singkatnya binari classification merupakan kegiatan mengkelasifikasi yang hanya memberikan dua class. contoh pada gambar 2.53 clasifikasi antara bentuk kotak dan segitiga.

2. Jelaskan Apaitu supervised learning , unsupervised learning dan clusterring dengan ilustrasi gambar sendiri.

supervised learning adalah cara untuk mengklasifikasikan suatu objek atau data yang telah di tentukan kelas kelasnya contoh pada sayuran tumbuhan wortel termasuk yang mengandung vitamin A berarti tumbuhan wortel telah dikategorikan kedalam sayuran yang mengandung vitamin A. sedangkan kangkung mengandung zat besi yang berarti tumbuhan kangkung telah dikategorikan kedalam sayuran yang mengandung zat besi untuk lebih jelasnya dapat dilihat pada gambar 2.54.

unsupervised learning merupakan cara untuk mengklasifikasi tanpa adanya kelas untuk menentukan jenisnya contoh sayuran berarti semua objek yang memiliki ciri ciri sayuran di kategorikan kedalam sayuran untuk lebih jelasnya dapat dilihat pada gambar 2.55.

clustering merupakan peroses mengklasifikasikan yang berdasarkan suatu parameter dalam penentuannya contoh pada berat sayuran sayuran A memiliki berat 100 gr dan sayuran B memiliki berat 120 gr yang berarti berat sayuran dibagi dua parameter yaitu lebih kecil samadengan 100 gram dan lebih besar dari gram contoh pada gambar 2.56.

3. Jelaskan apa itu evaluasi dan akurasi dan disertai ilustrasi contoh dengan gambar sendiri.

evaluasi adalah pengumpulan pengumpulan dan pengamatan dari berbagai macam bukti untuk mengukur dampak efektifitas dari suatu objek, program, atau proses berkaitan dengan spesifikasi atau persyaratan yang telah di tetapkan sebelumnya. sedangkan akurasi itu sendiri merupakan bagian dari evaluasi yang merupakan ketepatan data terhadap suatu objek berdasarkan keriteria tertentu. kita dapat mengevaluasi seberapa baik model bekerja dengan mengukur akurasinya. ketepatan akan di definisikan sebagai presentase kasus yang di klasifikasikan dengan benar. hal ini berkaitan dengan confusion matrix pada materi selanjutnya. contoh evaluasi untuk membedakan burung dengan ayam terdapat parameter yaitu ukuran badan dan fungsi sayap pada hewan tersebut. lebih jelanya pada gambar 2.57 berikut:

4. Jelaskan bagaimana cara membuat Confusion Matrix, Buat confusion matrix sendiri.

Dalam pembuatan confusion matrix tentukan parameter atau objek yang akan di evaluasi contoh bunga melati , bunga mawar, dan bunga kenangan buat tabel dengan baris dan kolom berjumlah tiga kemudian tentukan nilai miring pada setiap kolom tersebut disini saya memberi nilai 30 dengan ketentuan setiap baris harus berisi nilai 30 nilai tersebut jika terbagi ke kolom lain maka jumlahnya harus bernilai 30 jika tidak berarti data tersebut tidak akurat. untuk lebih jelanya dapat dilihat pada gambar 2.58 berikut :

5. Jelaskan bagaimana K-fold cross validation bekerja dengan gambar ilustrasi contoh buatan sendiri. K-fold Cross Validation merupakan cara untuk melatih suatu mesin dimana di dalamnya terdapat data set yang dibagi menjadi dua

yaitu untuk data testing dan data training contoh 1000 data merupakan data set dan 200 data digunakan untuk data testing kemudian 800 datanya digunakan untuk data training dimana data training tersebut digunakan untuk menentukan nilai bobot yang dimasukan kedalam rumus regresi linier. sedangkan nilai testing akan dijadikan nilai inputan untuk rumus regresi linier. contohnya dapat dilihat pada gambar 2.59 berikut :

6. Jelaskan Apa itu decision tree dengan gambar ilustrasi contoh buatan sendiri.

Decision tree (pohon keputusan) merupakan implementasi dari binari clasification dimana pada pohon keputusan akan terdapat root atau akar dan cabang cabangnya yang nilainya seperti if contoh pada root berisi nilai jenis kelamin, apakah perempuan pada cabang satu bernilai iya dan pada cabang dua bernilai tidak jika nilainya iya berarti jenis kelamminya perempuan dan jika tidak maka bernilai laki-laki. agar lebih jelas dapat dilihat pada gambar 2.60 decision tree berikut:

7. jelaskan apa itu information gain dan entropi dengan gambar ilustrasi buatan sendiri.

informasian gain merupakan informasi atau keriteria dalam pembagian sebuah objek contoh information gain pada laki-laki yaitu berrambut pendek, memiliki jakun, berjenggot, berkumis, dan mempunyai bahu yang lebar. pada kriteria tersebut seringkali terdapat bias misalkan ada perempuan yang berrambut pendek atau berkumis namun dari parameter tersebut dapat dilihat bahwa 60 persen parameter tersebut tepat pada sasarannya selama parameter itu bernilai tinggi untuk tepat maka dapat digunakan itulah information gain untuk lebih jelasnya dapat dilihat pada gambar 2.61 berikut :

sedangkan entropi merupakan ukuran keacakan dari informasi semakin tinggi entropi maka semakin sulit dalam menentukan keputusan contoh dalam menentukan jenis kelamin semakin detail informasi maka akan semakin susah dalam menentukan keputusan.

2.1.2 Sikic-Learn /Cokro Edi Prawiro/1164069

1. pada surcode pertama yang dapat dilihat pada gambar 2.1 pada baris pertama di tuliskan

```
import pandas as baso
```

yang berarti mengimport library padas yang di inisialisasi namanya menjadi baso. selanjutnya pada baris ke dua codingan tersebut berisi

```
cireng = baso.read_csv  
('E:\KULIAH\semester_6\AI\Buku\Chapter01\dataset\student-por.csv', sep=';')
```

pada code tersebut terdapat variabel cireng yang berisi inisialisasi padas (baso) dengan aksi untuk membaca vfile csv yang terdapat pada direktori pada komputer kemudian terdapat sep samadengan titik koma yang berarti pemisah field di dalam vile tersebut merupakan titik koma. kemudian pada bagian akhir terdapat code len (nama variabel) yang berarti akan menghotung jumlah baris pada file tersebut. untuk hasilnya dapat dilihat pada gambar 2.2

```
# In[1]:  
  
# Load dataset (student Portuguese scores)  
import pandas as baso  
cireng = baso.read_csv('E:\KULIAH\semester_6\AI\Buku\Chapter01\dataset\student-por.csv', sep=';')  
len(cireng)
```

Figure 2.1: Source Code Load Dataset

```
In [1]: import pandas as baso  
...: cireng = baso.read_csv('E:\KULIAH\semester_6\AI\Buku\Chapter01\dataset\student-por.csv', sep=';')  
...: len(cireng)  
Out[1]: 649
```

Figure 2.2: Hasil Load Dataset

2. Pada code selanjutnya akan ditambahkan fungsi untuk lulus atau gagal yang dibuat berupa kolom kolom yang di dalamnya terdapat nilai 0 dan 1 dimana 0 berarti gagal dan 1 berarti lulus. kemudian variabel pada codingan sebelumnya masih digunakan yaitu pada codingan berikut

```
cireng['pass'] = cireng.apply(lambda row:  
    1 if (row['G1']+row['G2']+row['G3']) >= 35 else 0, axis=1)
```

. dimana variabel cireng digunakan karena berisi nilai file csv kemudian dilakukan ekseskuji dengan parameter G1, G2, dan G3 dengan fungsi lambda yang berarti if bersarang atau if didalam if yang berarti nilai yang di eksekusi akan dilempar ke dalam setiap paramater sasuai dengan kriteria dan axis=1

yaitu nilai tersebut akan digunakan dari tiap baris data. sedangkan pada codingan

```
cireng = cireng.drop(['G1', 'G2', 'G3'], axis=1)
```

variabel cireng di berikan aksi drop yaitu penurunan nilai pada bagian baris data. dan pada code cireng.head () yaitu untuk mengeksekusi codingan sebelumnya untuk lebih jelasnya dapat dilihat pada gambar 2.3 dan hasilnya seperti pada gambar 2.4 berikut :

```
# In[2]:
# generate binary Label (pass/fail) based on G1+G2+G3 (test grades, each 0-20 pts); threshold for passing
cireng['pass'] = cireng.apply(lambda row: 1 if (row['G1']+row['G2']+row['G3']) >= 35 else 0, axis=1)
cireng = cireng.drop(['G1', 'G2', 'G3'], axis=1)
cireng.head()
```

Figure 2.3: memberikan nilai satu atau nol

```
In [2]: cireng['pass'] = cireng.apply(lambda row: 1 if (row['G1']+row['G2']+row['G3']) >= 35 else 0, axis=1)
...: cireng = cireng.drop(['G1', 'G2', 'G3'], axis=1)
...: cireng.head()
Out[2]:
   school sex  age address famsize ...  Dalc  Walc  health absences  pass
0       GP    F   18      U    GT3 ...     1     1      3        4      0
1       GP    F   17      U    GT3 ...     1     1      3        2      0
2       GP    F   15      U    LE3 ...     2     3      3        6      1
3       GP    F   15      U    GT3 ...     1     1      5        0      1
4       GP    F   16      U    GT3 ...     1     2      5        0      1
[5 rows x 31 columns]
```

Figure 2.4: hasil dari memberikan nilai nol dan satu

3. pada kodingan selanjutnya diguanakan untuk menambahkan nilai numerik berupa 0 dan 1 yang dirubah dari nilai tidak dan iya hal ini merupakan fungsi dari codingan get_dummies pada baris pertama pada gambar 2.5 yang nilainya diambil dari variabel cireng yang telah di dekralasikan tadi banyaknya data numerik itu sendiri tergantung pada field dari kolom yang di camtumkan pada codingan dengan catatan field tersebut harus ada dalam data csv yang di import oleh codingan pertama tadi maka hasilnya akan merubah data dalam field tersebut menjadi 0 dan 1 untuk lebih jelasnya dapat di lihat pada gambar 2.6 berikut;
4. selanjutnya pada codingan selanjutnya yaitu menentukan data training dan data testing dari dataset dimana variabel cireng yang berisi data csv tadi dibuat perbandingan 500 untuk data training dan sisanya yaitu 149 digunakan untuk

```
# In[3]:
# use one-hot encoding on categorical columns
cireng = baso.get_dummies(cireng, columns=['sex', 'school', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob',
                                             'reason', 'guardian', 'schoolsup', 'famsup', 'paid', 'activities',
                                             'nursery', 'higher', 'internet', 'romantic'])
cireng.head()
```

Figure 2.5: Penambahan nilai numerik

```
In [3]: cireng = baso.get_dummies(cireng, columns=['sex', 'school', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob',
...                                              'reason', 'guardian', 'schoolsup', 'famsup', 'paid', 'activities',
...                                              'nursery', 'higher', 'internet', 'romantic'])
Out[3]:
   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 2.6: hasil Penambahan nilai numerik

data testing hal ini di lakukan pada baris ke 1 sampai ke 3 pada gambar 2.7 kemudian nilai tersebut di turunkan brdasarkan baris pada data set hal itu dapat dilihat pada baris ke 4 sampai ke 9 pada gambar 2.7 kemudian selanjutnya mengimport library numpy yang berguna untuk oprasi vektor dan matrix karna data diatas berupa data matrix maka library ini di gunakan. untuk hasilnya dapat dilihat pada gambar 2.8 dan 2.9

```
# In[4]:
# shuffle rows
cireng = cireng.sample(frac=1)
# split training and testing data
cireng_train = cireng[:500]
cireng_test = cireng[500:]

cireng_train_att = cireng_train.drop(['pass'], axis=1)
cireng_train_pass = cireng_train['pass']

cireng_test_att = cireng_test.drop(['pass'], axis=1)
cireng_test_pass = cireng_test['pass']

cireng_att = cireng.drop(['pass'], axis=1)
cireng_pass = cireng['pass']

# number of passing students in whole dataset:
import numpy as nasipadang
print("Passing: %d out of %d (%.2f%%)" % (nasipadang.sum(cireng_pass), len(cireng_pass), 100*float(nasipadang.sum(cireng_pass)) / len(cireng_pass)))
```

Figure 2.7: penentuan data training dan data testing

5. selanjutnya yaitu membuat pohon keputusan dapat lebih jelasnya dapat di lihat pada gambar 2.10. pada baris pertama yairu memasukan librari tree kemudian pada baris kedua dibuat variabel tempe dengan nilai DecisionTreeClassifier yang merupakan paket atau fungsi dari scikit-learn yang merupakan class

```

In [4]: cireng = cireng.sample(frac=1)
.... # split training and testing data
.... cireng_train = cireng[:500]
.... cireng_test = cireng[500:]
.....
.... cireng_train_att = cireng_train.drop(['pass'], axis=1)
.... cireng_train_pass = cireng_train['pass']
.....
.... cireng_test_att = cireng_test.drop(['pass'], axis=1)
.... cireng_test_pass = cireng_test['pass']
.....
.... cireng_att = cireng.drop(['pass'], axis=1)
.... cireng_pass = cireng['pass']
.....
.... # number of passing students in whole dataset:
.... import numpy as nasipadang
.... print("Passing: %d out of %d (%.2f%%)" % (nasipadang.sum(cireng_pass), len(cireng_pass),
100*float(nasipadang.sum(cireng_pass)) / len(cireng_pass)))
Passing: 328 out of 649 (50.54%)

```

Figure 2.8: Hasil 1 penentuan data training dan data testing

| Name | Type | Size | Value |
|-------------------|-----------|-----------|---|
| cireng | DataFrame | (649, 57) | Column names: age, Medu, Fedu, travelttime, studytime, failures, famrel ... |
| cireng_att | DataFrame | (649, 56) | Column names: age, Medu, Fedu, travelttime, studytime, failures, famrel ... |
| cireng_pass | Series | (649,) | Series object of pandas.core.series module |
| cireng_test | DataFrame | (149, 57) | Column names: age, Medu, Fedu, travelttime, studytime, failures, famrel ... |
| cireng_test_att | DataFrame | (149, 56) | Column names: age, Medu, Fedu, travelttime, studytime, failures, famrel ... |
| cireng_test_pass | Series | (149,) | Series object of pandas.core.series module |
| cireng_train | DataFrame | (500, 57) | Column names: age, Medu, Fedu, travelttime, studytime, failures, famrel ... |
| cireng_train_att | DataFrame | (500, 56) | Column names: age, Medu, Fedu, travelttime, studytime, failures, famrel ... |
| cireng_train_pass | Series | (500,) | Series object of pandas.core.series module |

Figure 2.9: hasil 2 penentuan data training dan data testing

yang mampu melakukan multi class. sedangkan max_depth=5 merupakan untuk penyesuaian data terhadap pohon keputusan itu sendiri. untuk hasilnya dapat dilihat pada gambar 2.11

6. selanjutnya yaitu pembuatan gambar dari pohon keputusan yang tadi dibuat pada codingan sebelumnya pada baris pertama codingan ya itu mengimport library graphviz kemudian pada baris ke dua yaitu pemberian nilai pada variabel baru dot data nilainya diambil dari pembuatan pohon keputusan tadi kemudian ditentukannya nilai benar dan salah dari codingan tersebut setelah itu dibuatlah sebuah variabel untuk menampung hasil eksekusi tersebut kemudian variabel tersebut di running untuk lebih jelasnya dapat dilihat pada gambar 2.12 kemudian hasilnya dapat dilihat pada gambar 2.13.

```
# In[5]:  
  
# fit a decision tree  
from sklearn import tree  
tempe = tree.DecisionTreeClassifier(criterion="entropy", max_depth=5)  
tempe = tempe.fit(cireng_train_att, cireng_train_pass)
```

Figure 2.10: memberikan nilai pada pohon keputusan

```
In [5]: from sklearn import tree  
.... tempe = tree.DecisionTreeClassifier(criterion="entropy", max_depth=5)  
.... tempe = tempe.fit(cireng_train_att, cireng_train_pass)
```

Figure 2.11: hasil memberikan nilai pada pohon keputusan

7. selanjutnya pembuatan method untuk menyimpan data pohon dengan menarik data langsung dari pohon keputusan di buat tadi untuk code lebih jelasnya dapat dilihat pada gambar2.14. kemudian intuk hasilnya dapat dilihat pada gambar 2.15 berikut.
8. selanjutnya pada codingan berikut yaitu digunakan untuk mencetak nilai score rata-rata dari ketepatan data yang telah diolah tadi lebih jelasnya dapat dilihat pada gambar 2.16 kemudian untuk hasilnya dapat dilihat pada gambar 2.17 tersebut:
9. selanjutnya yaitu digunakan untuk memeriksa akurasi dari ketepatan hasil pengolahan data tersebut maka akan didapat nilai rata-rata 60 persen dari hasil pengolahan data tersebut untuk lebih jelasnya dapat dilihat pada gambar 2.18 pada codingan tersebut pada baris ke satu melakukan import library dari sklearn kemudian pada baris selanjutnya mengisi nilai skor dengan nilai pada variabel tempe setelah hal tersebut dilakukan kemudian data tersebut di eksekusi. berikut merupakan hasil dari code tersebut dapat dilihat pada gambar 2.19.
10. membuat rank akurasi dari 1 sampai 20 untuk melihat akurasi data apakah datatersebut terdapat di rata rata 60 persen atau tidak dengan cara membuat lagi variabel baru dengan nilai tree diadalamnya jadi hampir mirip seperti membuat pohon keputusan namun ini langsung dalam bentuk rata rata akurasi yanlebih spesifik untuk lebih jelasnya dapat dilihat pada gambar2.20 codingan berikut. dan untuk hasilnya dapat dilihat pada gambar 2.21 berikut.

```
# In[6]:  
  
# visualize tree  
import graphviz  
dot_data = tree.export_graphviz(temp, out_file=None, label="all", impurity=False, proportion=True,  
                                feature_names=list(cireng_train_att), class_names=["fail", "pass"],  
                                filled=True, rounded=True)  
graph = graphviz.Source(dot_data)  
graph
```

Figure 2.12: pembuatan diagram pohon keputusan

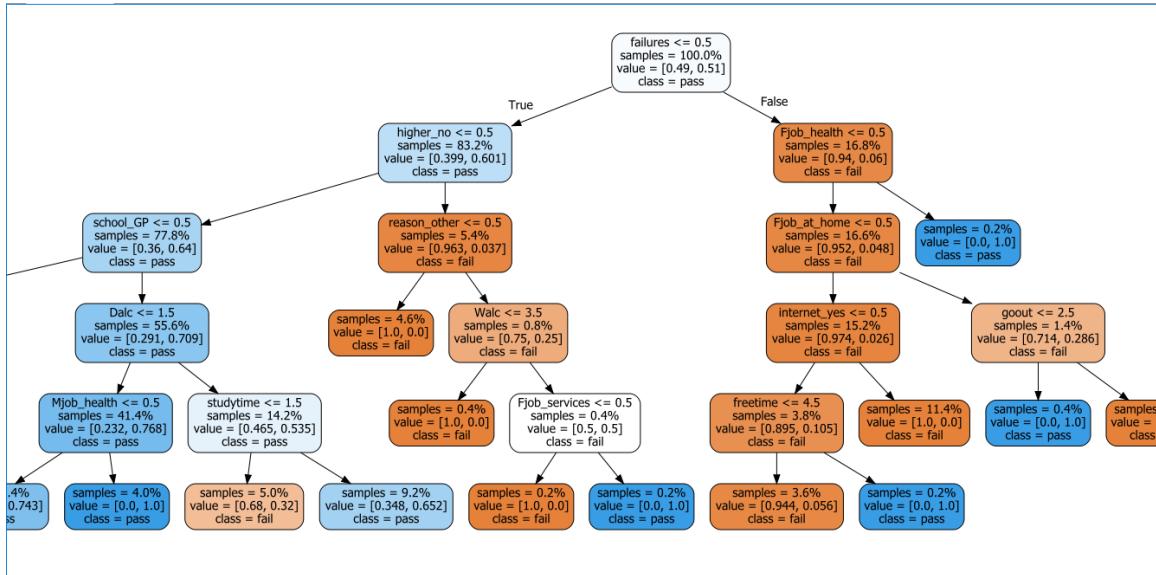


Figure 2.13: hasil pembuatan pohon keputusan

11. untuk selanjutnya yaitu menentukan nilai untuk grafik hampir sama dengan nilai akurasi tadi pertama tentukan rank atau batasan nilai itu disini batasannya dimulai dari 1 sampai 20 dengan menggunakan nilai tree tadi maka hasilnya dapat ditentuka kemudian buat variabel i untuk penomoran tiap record yang keluar atau record hadil dari eksekusi tree tersebut. untuk lebih jelasnya dapat dilihat pada gambar 2.22 dan untuk hasilnya dapat dilihat pada gambar 2.23.
12. terakhir yaitu pembuatan grafik untuk pembuatan grafik diambil data dari codingan sebelumnya yang 20 record tadi dengan cara mengimport library matplotlib.pyplot yang diinisialisasi menjadi bawankemudian inisialisasi tersebut di eksekusi. untuk lebih jelasnya codingannya seperti gambar 2.24 dan untuk hasilnya seperti gambar 2.25 berikut.

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

Figure 2.14: coding save

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

Figure 2.15: hasil coding save

2.1.3 Penanganan Error /Cokro Edi Prawiro/1164069

1. skrinsut error dapat dilihat pada gambar 2.26
2. kode error dan jenis errornya .

```
import graphviz  
dot_data = tree.export_graphviz(tree, out_file=None, label="all", impurity=False,  
                                feature_names=list(cireng_train_att), class_names=["fail", "pass"],  
                                filled=True, rounded=True)  
graph = graphviz.Source(dot_data)  
graph
```

pada codingan tersebut error karena graphviznya belum di install

3. Solusi pemecahan masalah

buka CMD komputer anda run as administrator kemudian masukan perintah conda install graphviz kemudian tekan enter ingat hal ini dilakukan harus terkoneksi dengan jaringan internet. untuk hasilnya dapat dilihat pada gambar 2.27 dan gambar 2.28 setelah itu masukan PATH graphviz dengan cara masuk ke direktori graphviz itu di simpan kalau di komputer saya disimpan di

C:\ProgramData\Anaconda3\Library\bin\graphviz

kemudian setelah itu copy alamat direktori tersebut dan masukan kedalam path seperti pada gambar 2.29 dan pada gambar 2.30.

```
# In[8]:  
  
tempe.score(cireng_test_att, cireng_test_pass)
```

Figure 2.16: Contoh Binary Classification

```
In [8]: tempe.score(cireng_test_att, cireng_test_pass)  
Out[8]: 0.6577181208053692
```

Figure 2.17: hasil coding save

2.2 Ahmad Syafrizal Huda/1164062

2.2.1 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 memisahkannya menjadi dua kelas, yaitu pena dan pensil.

Contoh ilustrasi gambar bisa dilihat pada gambar 2.45.

2. Supervised learning merupakan sebuah pendekatan dimana sudah terdapat data yang dilatih, dan terdapat variable yang ditargetkan sehingga tujuan dari pendekatan ini adalah mengelompokan suatu data ke data yang sudah ada. Sedangkan unsupervised learning tidak memiliki data latih, sehingga dari data yang ada, kita mengelompokan 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[6].

Contoh ilustrasi gambar bisa dilihat pada gambar 2.46.

Contoh ilustrasi gambar bisa dilihat pada gambar 2.47.

```
# In[9]:
from sklearn.model_selection import cross_val_score
scores = cross_val_score(tempe, cireng_att, cireng_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))
```

Figure 2.18: Akurasi perhitungan pohon keputusan

```
In [9]: from sklearn.model_selection import cross_val_score
.... scores = cross_val_score(tempe, cireng_att, cireng_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: 0.71 (+/- 0.06)
```

Figure 2.19: hasil Akurasi perhitungan pohon keputusan

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

Contoh ilustrasi gambar bisa dilihat pada gambar 2.49.

- 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.

```
Recall =3/(1+3) = 0,75
Precision = 3/(1+3) = 0,75
Accuracy =(5+3)/(5+1+1+3) = 0,8
Error Rate =(1+1)/(5+1+1+3) = 0,2
```

- 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.

```
# In[10]:  
  
for max_depth in range(1, 20):  
    tempeenak = tree.DecisionTreeClassifier(criterion="entropy", max_depth=max_depth)  
    scores = cross_val_score(tempeenak, cireng_att, cireng_pass, cv=5)  
    print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" % (max_depth, scores.mean(), scores.std() * 2))
```

Figure 2.20: Contoh Binary Classification

```
In [10]: for max_depth in range(1, 20):  
....:     tempeenak = tree.DecisionTreeClassifier(criterion="entropy", max_depth=max_depth)  
....:     scores = cross_val_score(tempeenak, cireng_att, cireng_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 (+/- 0.05)  
Max depth: 2, Accuracy: 0.69 (+/- 0.07)  
Max depth: 3, Accuracy: 0.70 (+/- 0.10)  
Max depth: 4, Accuracy: 0.70 (+/- 0.07)  
Max depth: 5, Accuracy: 0.71 (+/- 0.06)  
Max depth: 6, Accuracy: 0.69 (+/- 0.08)  
Max depth: 7, Accuracy: 0.68 (+/- 0.04)  
Max depth: 8, Accuracy: 0.70 (+/- 0.07)  
Max depth: 9, Accuracy: 0.69 (+/- 0.07)  
Max depth: 10, Accuracy: 0.68 (+/- 0.09)  
Max depth: 11, Accuracy: 0.68 (+/- 0.09)  
Max depth: 12, Accuracy: 0.68 (+/- 0.07)  
Max depth: 13, Accuracy: 0.67 (+/- 0.06)
```

Figure 2.21: hasil Akurasi perhitungan pohon keputusan

- 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 2.50.

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 2.51.

7. Gain adalah pengurangan yang diharapkan dalam entropy. Dalam machine learning, gain dapat digunakan untuk menentukan sebuah urutan atribut atau

```
# In[11]:\n\ndepth_acc = nasipadang.empty((19,3), float)\n\ni = 0\nfor max_depth in range(1, 20):\n    tempemendoan = tree.DecisionTreeClassifier(criterion="entropy", max_depth=max_depth)\n    scores = cross_val_score(tempemendoan, cireng_att, cireng_pass, cv=5)\n    depth_acc[i,0] = max_depth\n    depth_acc[i,1] = scores.mean()\n    depth_acc[i,2] = scores.std() * 2\n    i += 1\n\ndepth_acc
```

Figure 2.22: Contoh Binary Classification

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 di artikan entropi adalah ukuran ketidakpastian dari sebuah atribut.

Contoh ilustrasi gambar bisa dilihat pada gambar 2.52.

2.2.2 Scikit-learn

- 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 pemisahnya 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 2.31.

- 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 terdefinisikan angka 1 apabila tidak, maka akan terdefinisikan 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 terubah sesuai dengan baris yang dieksekusi.

```

In [11]: depth_acc = nasipadang.empty((19,3), float)
....: i = 0
....: for max_depth in range(1, 20):
....:     tempemendoan = tree.DecisionTreeClassifier(criterion="entropy", max_depth=max_depth)
....:     scores = cross_val_score(tempemendoan, cireng_att, cireng_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[11]:
array([[ 1.        ,  0.63779741,  0.05447395],
       [ 2.        ,  0.68706505,  0.07197636],
       [ 3.        ,  0.69774069,  0.09857052],
       [ 4.        ,  0.70088863,  0.07289402],
       [ 5.        ,  0.71017885,  0.05597996],
       [ 6.        ,  0.68705331,  0.08098517],
       [ 7.        ,  0.68095891,  0.03757078],
       [ 8.        ,  0.69627251,  0.07383519],
       [ 9.        ,  0.68700597,  0.07922729],
      [10.        ,  0.67468617,  0.07965841],
      [11.        ,  0.67927752,  0.10138467],
      [12.        ,  0.69017812,  0.05085986],
      [13.        ,  0.66856774,  0.05944621],
      [14.        ,  0.65625968,  0.05003322],
      [15.        ,  0.66554953,  0.07030686],
      [16.        ,  0.6748403 ,  0.02486122],
      [17.        ,  0.66244894,  0.05728159],
      [18.        ,  0.65016492,  0.04288463],
      [19.        ,  0.66713588,  0.03161508]])

```

Figure 2.23: hasil Akurasi perhitungan pohon keputusan

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.32.

3. Penjelasan codingan ini mendefinisikan pemanggilan get dummies dari buah-naga dalam variabel buahpir. Di dalam get dummies sendiri akan terdefinisikan 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 2.33.

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 pengeksekusian untuk kolom Pass, apabila sesuai dengan axis=1 maka eksekusi

```
# In[12]:  
  
import matplotlib.pyplot as bakwan  
fig, ax = bakwan.subplots()  
ax.errorbar(depth_acc[:,0], depth_acc[:,1], yerr=depth_acc[:,2])  
bakwan.show()
```

Figure 2.24: codingan pembuatan grafik

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 2.34.

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 mengimport 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 train att dan buahpir train pass.

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.35.

6. Penjelasan codingan ini memberikan gambaran dari klasifikasi decision tree 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 2.36.

7. 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 tree nya dapat berjalan atau tidak. Apabila tidak berjalan, maka akan terjadi error, namun codingan ini berfungsi.

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.37.

8. 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

```
In [12]: import matplotlib.pyplot as bakwan
....: fig, ax = bakwan.subplots()
....: ax.errorbar(depth_acc[:,0], depth_acc[:,1], yerr=depth_acc[:,2])
....: bakwan.show()
```

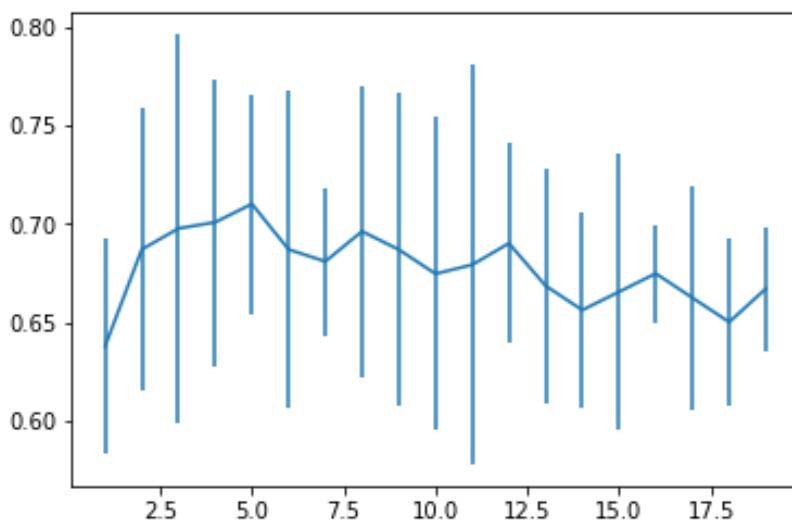


Figure 2.25: hasil grafik

yang dieksekusi memang memiliki data sehingga dieksekusi dan menghasilkan keluaran dari score tersebut.

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.38.

9. Penjelasan codingan ini membahas mengenai pengeksekusian 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 ditampilkan ialah rata-rata perhitungan dari variabel score dimana dan standar dari plus minusnya tentunya dengan ketentuan parameter Accuracy .

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.39.

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

```

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

File "<ipython-input-21-461c085f0565>", line 1, in <module>
    import graphviz

ModuleNotFoundError: No module named 'graphviz'

```

Figure 2.26: Screensot error

ditampilkan ialah dari max depth, accuracy dan plus minusnya dan akhirnya hasil outputannya keluar.

Gambar Screenshootan codingan dan hasil bisa dilihat pada gambar 2.40.

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 2.41.

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 2.42.

2.2.3 Penanganan Eror

1. ScreeShootan Eror pada codingan No 8 dapat dilihat pada gambar 2.43.

```

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

## Package Plan ##

environment location: C:\ProgramData\Anaconda3

added / updated specs:
- graphviz

The following packages will be downloaded:

  package          |      build
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)? y

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

EnvironmentNotWritableError: The current user does not have write permissions to the target environment.
  environment location: C:\ProgramData\Anaconda3

C:\Users\COKRO>

```

Figure 2.27: Proses instalasi graphviz

2. Codingan eror dan jenis erornya : sebenarnya tidak terdapat eror pada codingan ini namun saat pertama kali di run current cell codingan ini akan eror 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=False,
                                feature_names=list(buahpir_train_att), class_names=True,
                                filled=True, rounded=True)
graph = graphviz.Source(dot_data)
graph

```

3. Solusi pemecahan masalah eror tersebut yaitu dengan cara menginstall terlebih dahulu library graphviznya pada anaconda prompt atau command prompt anda dengan perintah conda install graphviz setelah itu run kembali codingan No 8 maka akan muncul outputan atau tampilan keluarannya.

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

C:\WINDOWS\system32>conda install graphviz
Collecting package metadata: done
Solving environment: done

## Package Plan ##

environment location: C:\ProgramData\Anaconda3

added / updated specs:
- graphviz

The following NEW packages will be INSTALLED:

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

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

C:\WINDOWS\system32>
```

Figure 2.28: Proses instalasi graphviz di user

Berikut gambar cara menginstall graphviz dapat dilihat pada gambar 2.44

Contoh ilustrasi gambar bisa dilihat pada gambar 2.48.

2.3 Fathi Rabbani / 1164074

2.3.1 Teori

1. Binary Classification

membuat sebuah klasifikasi dengan menggunakan 2 buah hasil data yang menghasilkan himpunan data dalam dua kelompok yang berbeda. berikut adalah contohnya 2.72.

2. Supervised, Unsupervised and Clustering

- Supervised Learning supervised learning adalah cara untuk mengklasifikasikan suatu objek atau data yang telah di tentukan kelasnya, berikut adalah contohnya 2.73.
- Unsupervised Learning unsupervised learning merupakan cara untuk mengklasifikasi tanpa adanya kelas untuk menentukan jenis datanya, berikut ini contohnya 2.74.

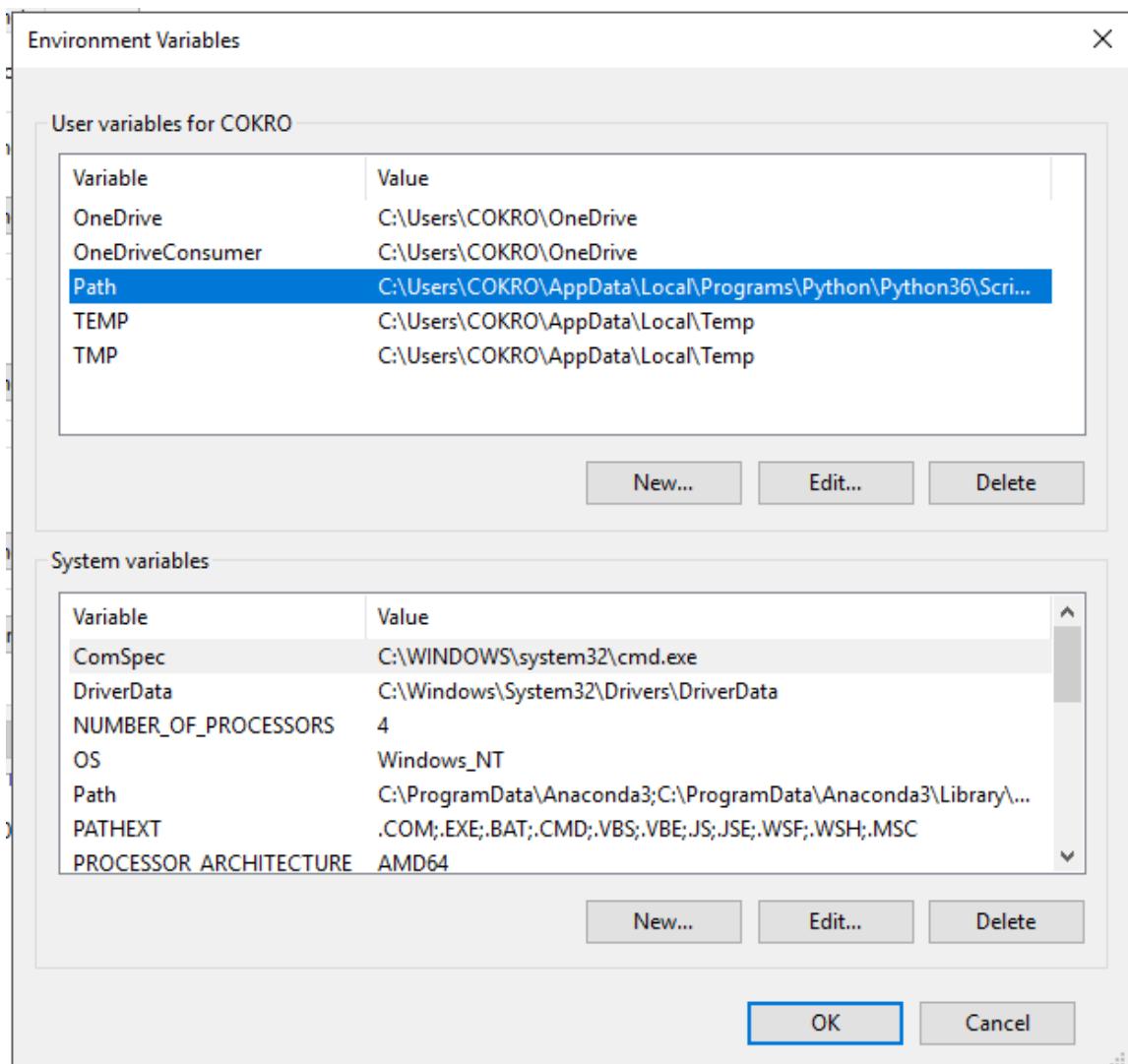


Figure 2.29: Path Komputer

- Clustering clustering merupakan peroses mengklasifikasikan yang berdasarkan suatu parameter dalam penentuan hasilnya, berikut contohnya 2.75

3. Evaluasi dan Akurasi

- Evaluasi evaluasi adalah sebuah proses dalam mengumpulkan serta mengamati bukti untuk mengukur dampak dari suatu objek, data, program atau proses yang berkaitan itu sendiri, berikut contohnya 2.76.
- Akurasi akurasi merupakan ketepatan dalam sebuah proses dalam melakukan perhitungan akan proses yang sedang berlangsung.

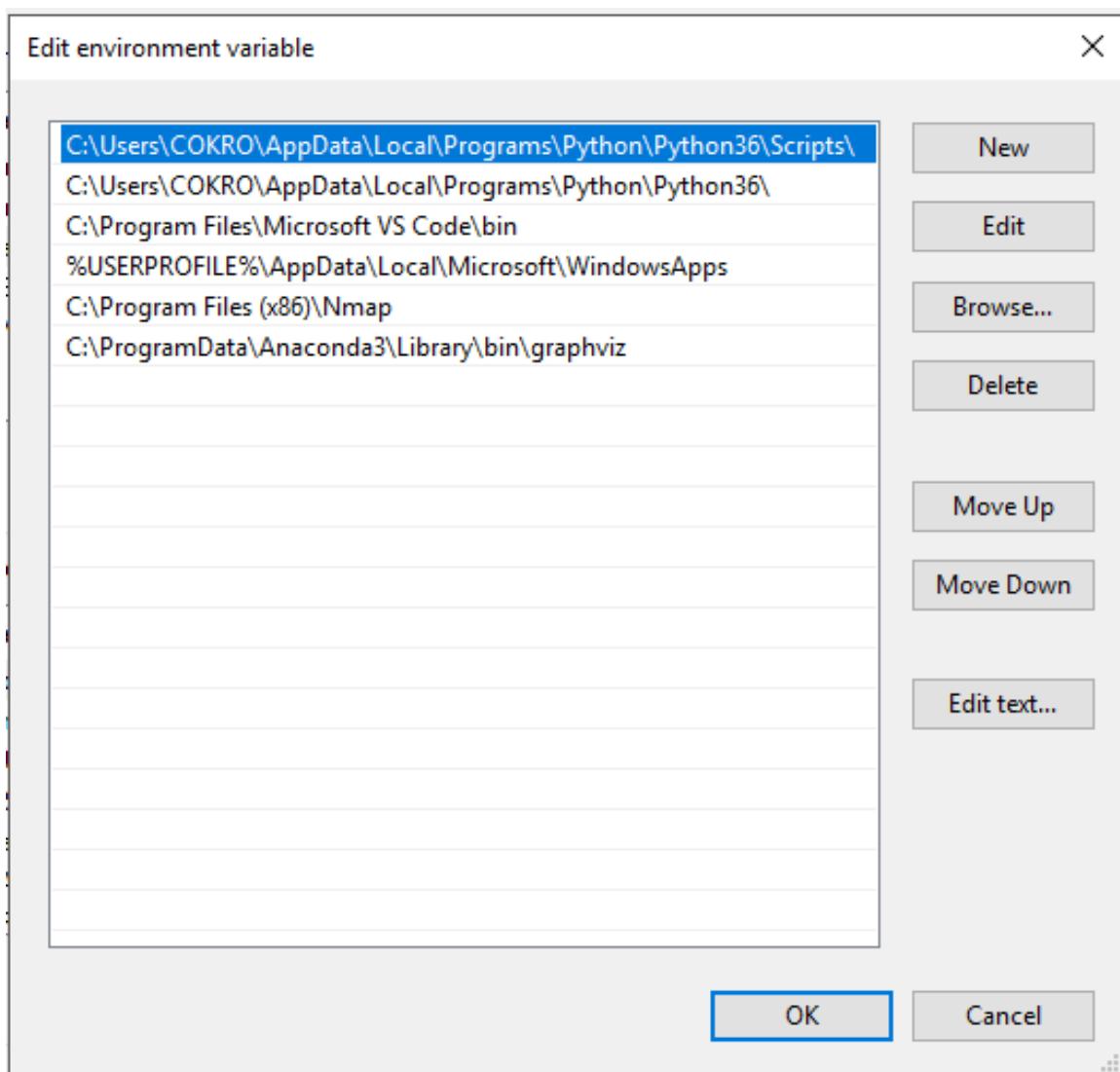


Figure 2.30: Memasukan Direktori Ke Path

4. Membuat dan Membaca Confusion Matrix menentukan objek yang digunakan sebagai bahan uji, sebagai contoh usia 20, 30 dan 40 dengan membuat sebuah tabel yang dapat menampung data dengan nilai 10 pada gambar 2.77 . lalu data tersebut bisa juga berupa data sebagai berikut 2.78. membaca data Matrix tersebut dengan menggunakan Usia sebagai data standarnya dengan rentang usia 20 hingga 40 tahun, lalu jumlah orang yang dapat di ketahui adalah 10 dan data yang ternilai haruslah berisi 10 agar data tersebut valid.
5. K-Fold Cross Validation K-fold Cross Validation merupakan cara untuk melatih suatu mesin dimana di dalamnya terdapat data set yang dibagi menjadi dua yaitu untuk data testing dan data training contoh 100 database besar 20 data digu-

```
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 2.31: Hasil Codingan No 1.

```
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 2.32: Hasil Codingan No 2.

nakan untuk data testing kemudian 80 datanya digunakan untuk data training dimana data training tersebut digunakan untuk menentukan nilai bobot yang dimasukan kedalam rumus regresi linier. seperti berikut 2.79.

6. Decision Tree merupakan implementasi dari binari clasification dimana akan terdapat akar dan cabang data yang memiliki nilai if...else contoh pada akar data berisi nilai jenis kelamin, apakah pria pada cabang satu bernilai iya dan pada cabang dua bernilai tidak jika nilainya iya berarti jenis kelaminya pria dan jika tidak maka bernilai wanita, lebih jelasnya dapat dilihat pada gambar2.80.
7. Information Gain dan Entropi informasion gain proses dengan mempraktekkan sistem decision tree menggunakan prinsip if...else yang berlangsung hingga menghasilkan data yang diinginkan. untuk contohnya seperti berikut ini sedangkan entropi merupakan ukuran keacakan dari informasi. semakin tinggi entropi maka semakin sulit dalam menentukan keputusan, berikut contohnya 2.81.

2.3.2 Praktek

- Code

1.

```
import pandas as plum
durian = plum.read_csv('dataset/student-mat.csv', sep=';')
```

```

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 2.33: Hasil Codingan No 3.

```
len(durian)
```

pada code berikut menjelaskan data dari pandas dengan menggunakan nama alias plum yang akan membaca data student-mat.csv yang berada pada folder dataset. hasilnya terdapat pada gambar 2.72

2.

```
durian['pass'] = durian.apply(lambda row: 1 if (row['G1']+row['G2']+row['G3'])>15 else 0)
durian = durian.drop(['G1', 'G2', 'G3'], axis=1)
durian.head()
```

pada code berikut ini menjelaskan bahwa slot data pada student-mat.csv akan ditambah dengan kolom pass dan menggunakan proses lambda yang akan menghasilkan data berupa array dengan struktur G1, G2 dan G3. hasilnya bisa dilihat pada gambar 2.73

3.

```
durian = plum.get_dummies(durian, columns=['sex', 'school', 'address',
                                              'famsup', 'famsize',
                                              'Pstatus', 'Medu', 'Fedu',
                                              'traveltime', 'studytime',
                                              'failures', 'internet',
                                              'nursery', 'higher', 'romantic'])
durian.head()
```

pada code berikut menerangkan bahwa variable durian memiliki proses yang akan memanggil variable plum untuk mendapatkan data pada kolom tersebut. hasilnya adalah 2.74

4.

```
durian = durian.sample(frac=1)
# split training and testing data
d_train = durian[:500]
```

```
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 2.34: Hasil Codingan No 4.

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

Figure 2.35: Hasil Codingan No 5.

```
d_test = durian[500:]

d_train_att = d_train.drop(['pass'], axis=1)
d_train_pass = d_train['pass']

d_test_att = d_test.drop(['pass'], axis=1)
d_test_pass = d_test['pass']

d_att = durian.drop(['pass'], axis=1)
d_pass = durian['pass']

# number of passing students in whole dataset:
import numpy as nanas
print("Passing: %d out of %d (%.2f%%)" % (nanas.sum(d_pass), len(d_pass)))
```

code berikut menjelaskan bahwa data durian akan diproses untuk di-

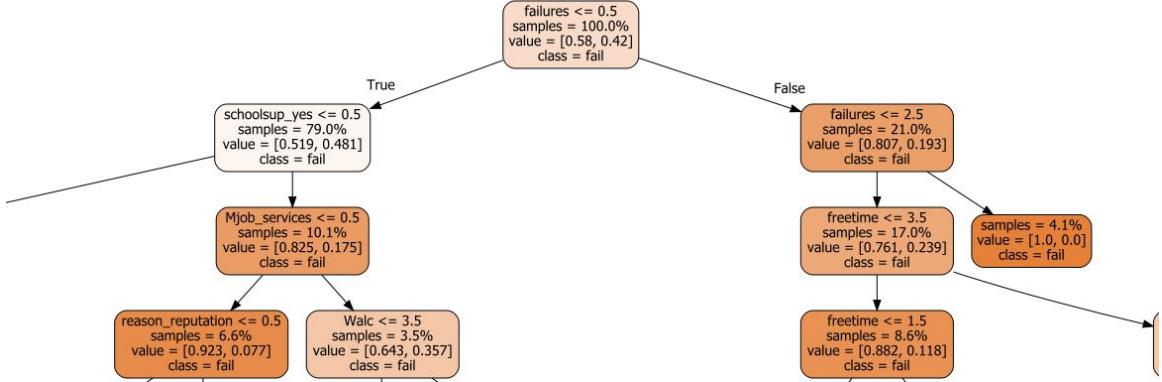


Figure 2.36: Hasil Codingan No 6.

```
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 2.37: Hasil Codingan No 7.

dapatkan hasil dari penggunaan k-fold cross yang membagi data dengan training dan testing dengan hasilnya adalah seperti pada gambar 2.75

```
5. from sklearn import tree
tomat = tree.DecisionTreeClassifier(criterion="entropy", max_depth=5)
tomat = tomat.fit(d_train_att, d_train_pass)
```

code ini mengambil data dari sklearn berupa data tree dengan variable tomat yang menampung data proses penggunaan deciontree dan di proses dengan menggunakan data d_train_att dan d_train_pass hasilnya ada digambar 2.76

```
6. import graphviz
dot_data = tree.export_graphviz(tomat, out_file=None, label="all", impur...
...:                         feature_names=list(d_train_att), class_na...
...:                         filled=True, rounded=True)

graph = graphviz.Source(dot_data)
graph
```

pada code berikut menjelaskan data graphviz yang diambil untuk menampilkan data 2.77

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

Figure 2.38: Hasil Codingan No 8.

```
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 2.39: Hasil Codingan No 9.

```
7. tree.export_graphviz(tomat, out_file="student-performance.dot", label="all",
feature_names=list(d_train_att), class_names=["fail", "pass"],
filled=True, rounded=True)
```

pada code ini digunakan untuk memproses data pada code 6 yang akan menghasilkan sebuah file bernama student-performance.dot hasilnya ada pada gambar 2.78

```
8. tomat.score(d_test_att, d_test_pass)
```

pada code ini dihasilkan data seperti berikut 2.79 yang artinya adalah data score dari d_test_att dan d_test_pass

```
9. from sklearn.model_selection import cross_val_score
apel = cross_val_score(tomat, d_att, d_pass, cv=5)
```

```
print("Accuracy: %0.2f (+/- %0.2f)" % (apel.mean(), apel.std() * 2))
```

mengambil data dari sklearn.model_selection dan mengambil data cross_val_score yang digunakan untuk menghitung data akurasi dari code 5 hasilnya ada digambar 2.80

```
10. for max_depth in range(1, 20):
    tomat = tree.DecisionTreeClassifier(criterion="entropy", max_depth=max_depth)
    apel = cross_val_score(tomat, d_att, d_pass, cv=5)
    print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" % (max_depth, apel.mean(),
    apel.std() * 2))
```

```
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: %.2f (+/- %.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 2.40: Hasil Codingan No 10.

code ini menjelaskan hasil dari akurasi pada code 9 yang dibreakdown untuk tampil sebagai data yang terhitung seperti pada gambar 2.81

```
11. semangka = nanas.empty((19,3), float)
    i = 0
    for max_depth in range(1, 20):
        tomat = tree.DecisionTreeClassifier(criterion="entropy", max_depth=max_depth)
        apel = cross_val_score(tomat, d_att, d_pass, cv=5)
        semangka[i,0] = max_depth
        semangka[i,1] = apel.mean()
        semangka[i,2] = apel.std() * 2
        i += 1

semangka
```

pada code ini data pada code 10 di ubah menjadi seurutan array yang ada pada gambar 2.82

```
12. import matplotlib.pyplot as melon
    fig, ax = melon.subplots()
    ax.errorbar(semangka[:,0], semangka[:,1], yerr=semangka[:,2])
    melon.show()
```

```

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 2.41: Hasil Codingan No 11.

code ini menampilkan data grafik yang digunakan untuk melihat data pada code sebelumnya hasilnya adalah seperti pada gambar 2.83

- Hasil

2.3.3 Penanganan Error

- Error Path

cara membenarkan error yang terdapat pada gambar 2.84 adalah dengan menginstall ulang graphviz dengan format

`conda install graphviz`

atau

`pip install graphviz`

yang terdapat pada gambar 2.85

```
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()
```

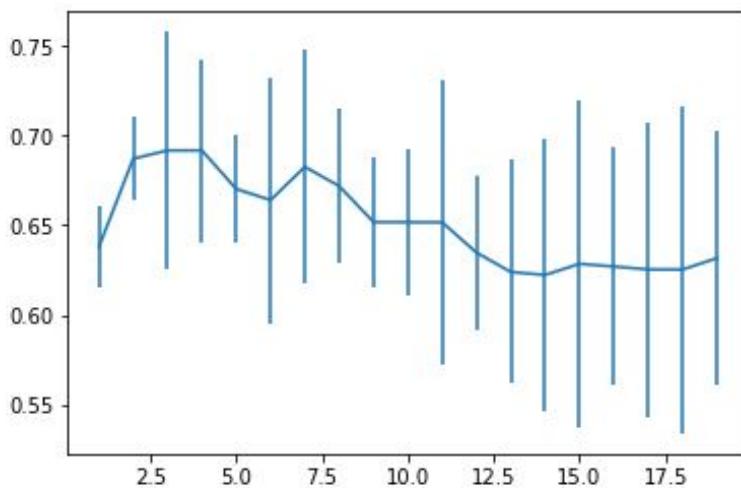


Figure 2.42: Hasil Codingan No 12.

```
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 2.43: Hasil Gambar Eror 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
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)? y

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

Figure 2.44: Hasil Gambar Penanganan Eror No 6.

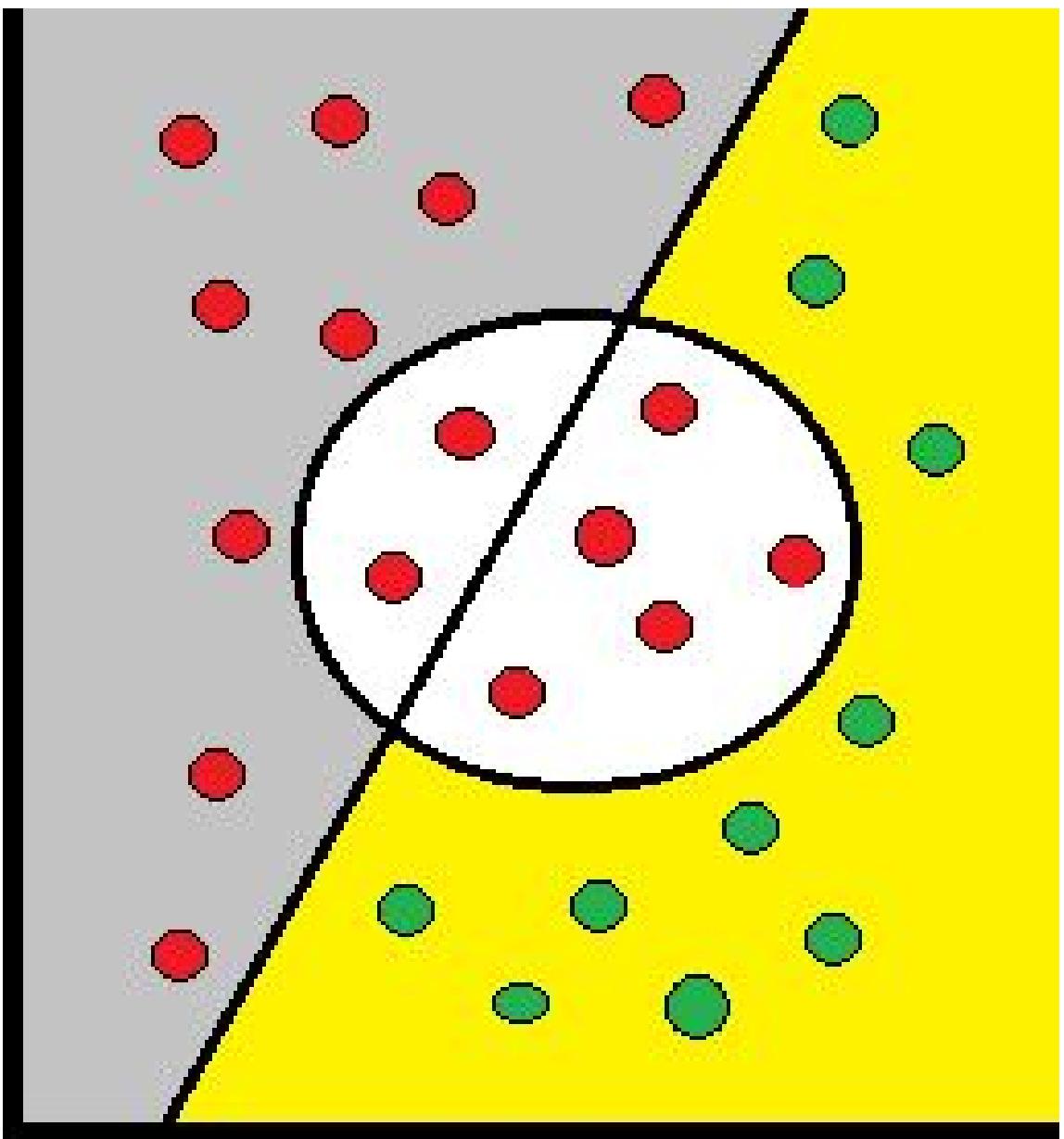


Figure 2.45: Binary Classification.

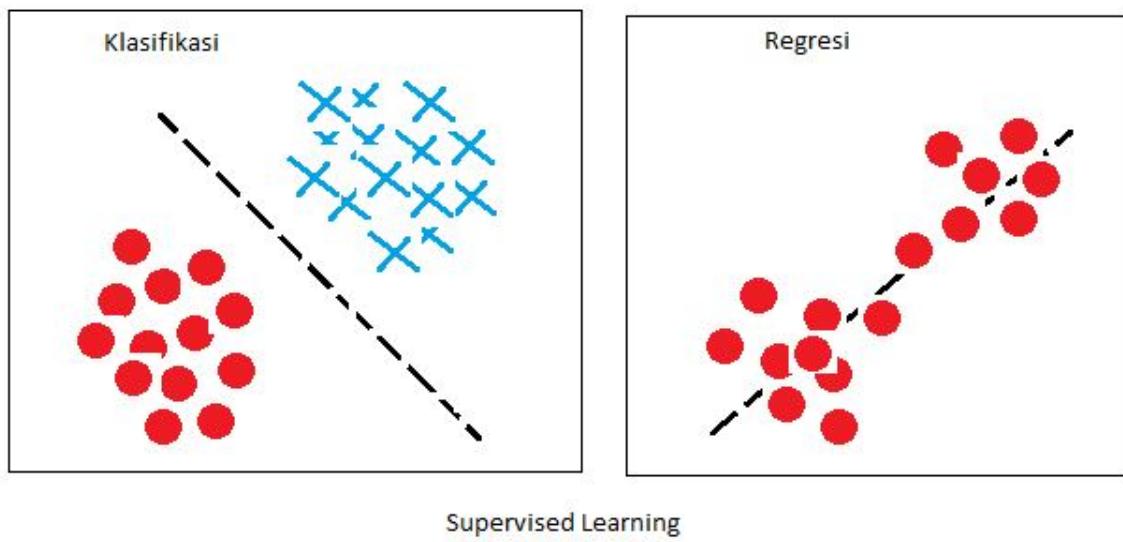


Figure 2.46: Supervised Learning.

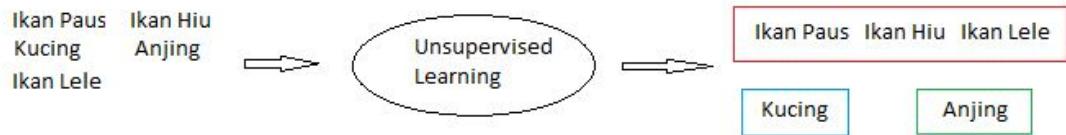


Figure 2.47: Unsupervised Learning.

Clustering

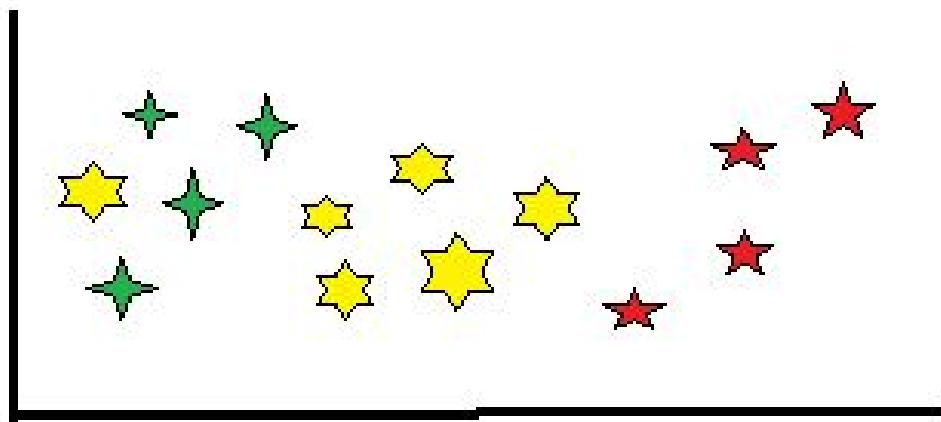


Figure 2.48: Clustering.

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

Figure 2.49: Evaluasi dan Akurasi.

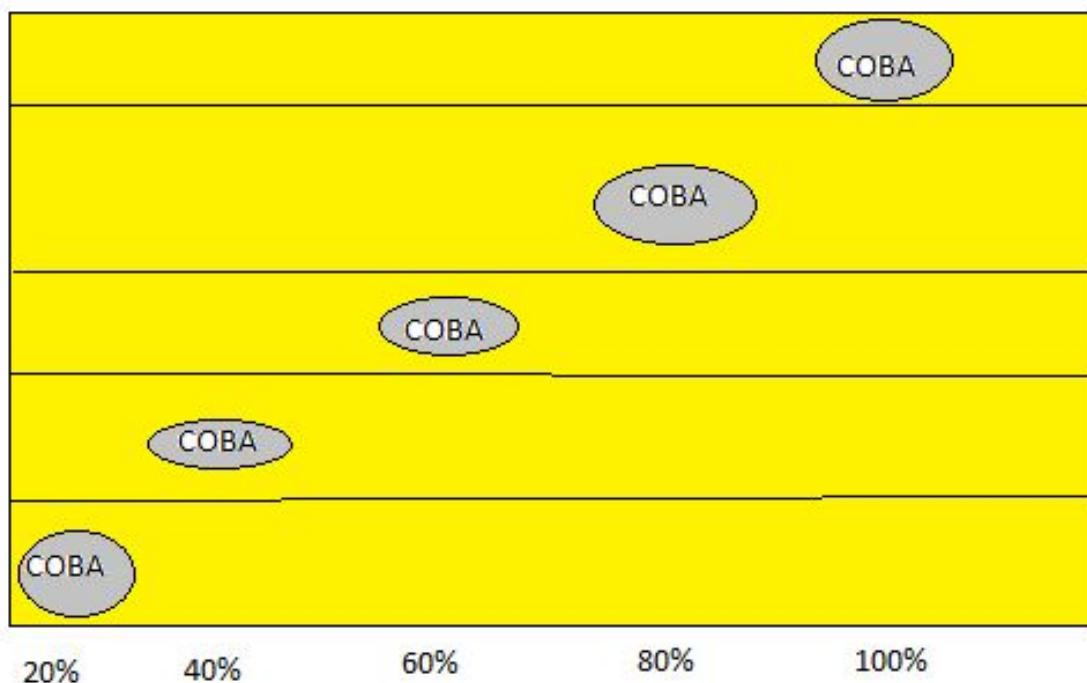


Figure 2.50: K-fold Cross Validation.

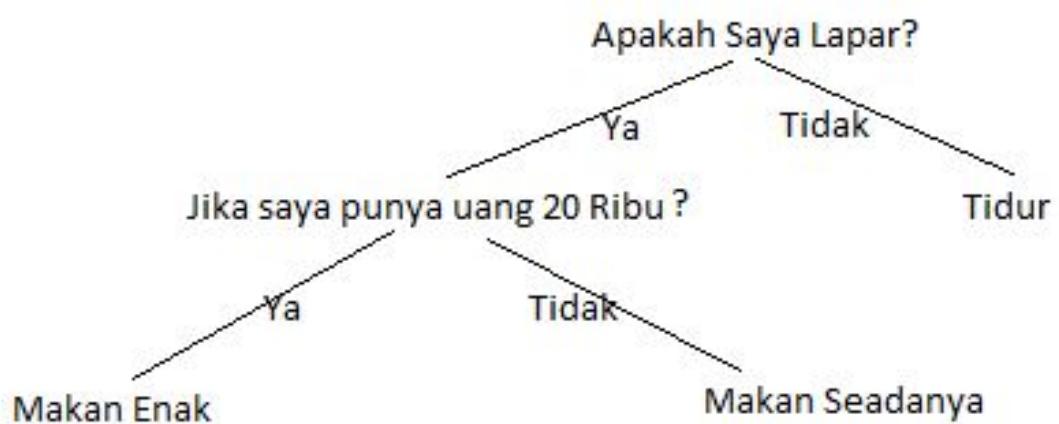


Figure 2.51: Decision Tree.

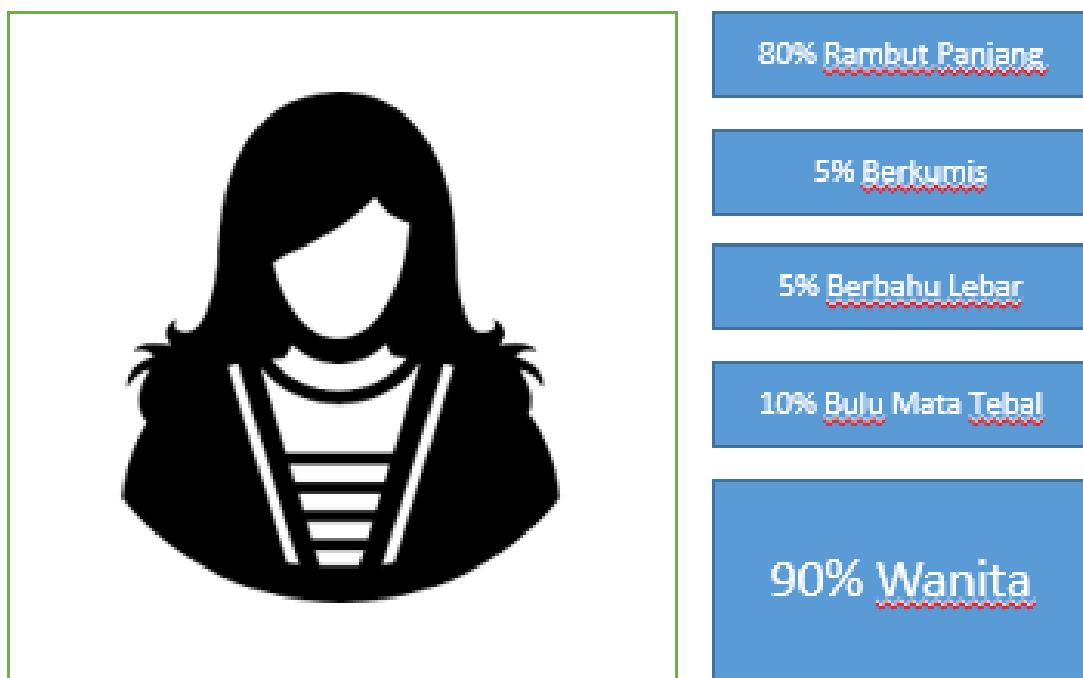
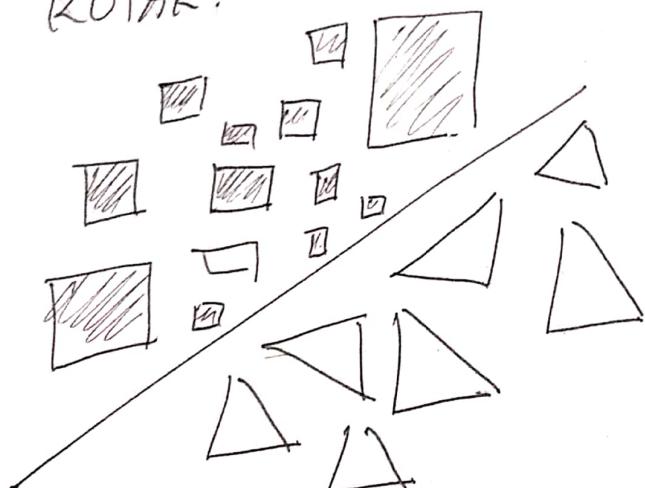


Figure 2.52: Gain.

Binary Classification.

KOTAK.



SEGITIGA

Figure 2.53: Contoh Binary Classification

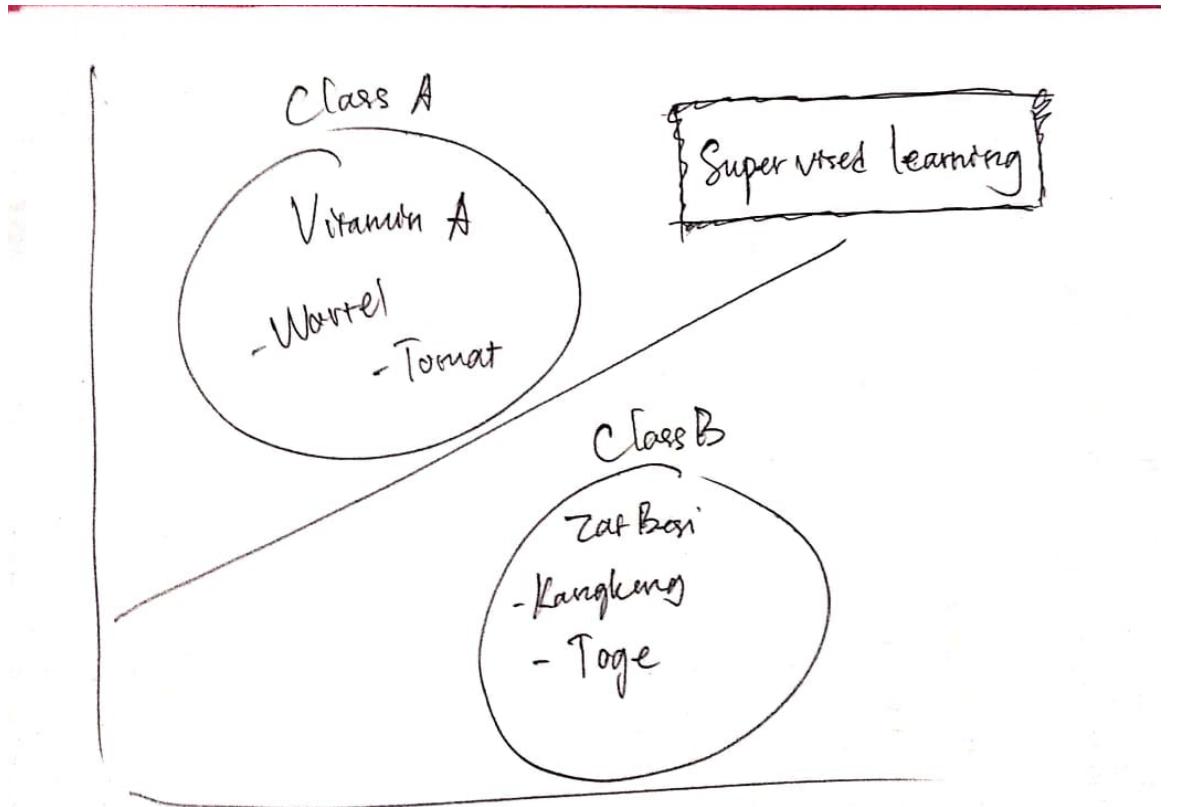


Figure 2.54: Ilustrasi Suervised Learning

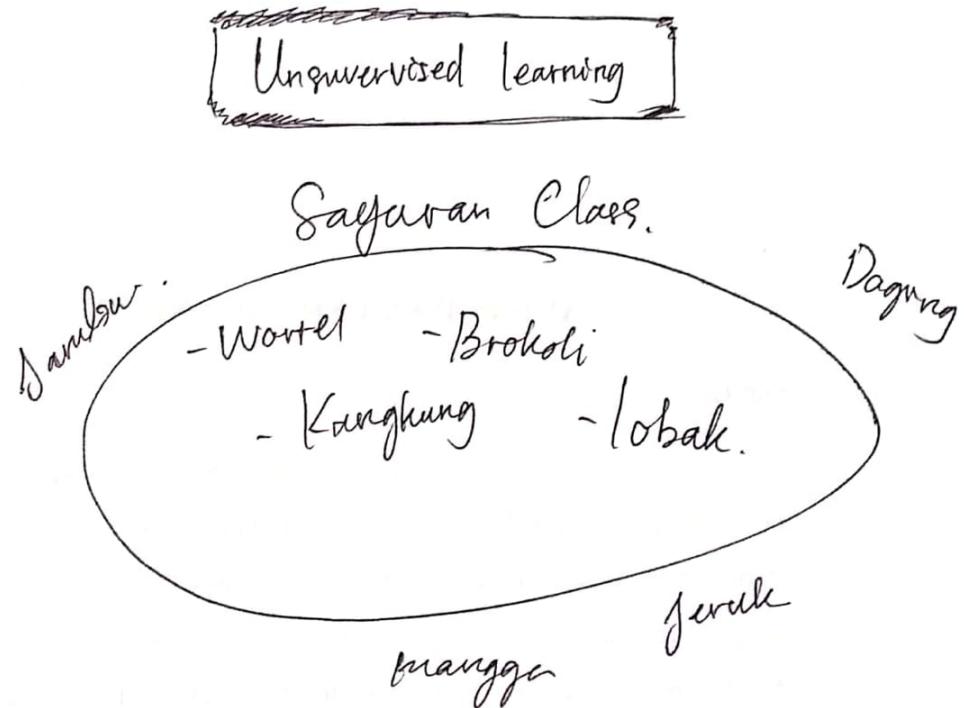


Figure 2.55: Ilustrasi Unsuervised Learning

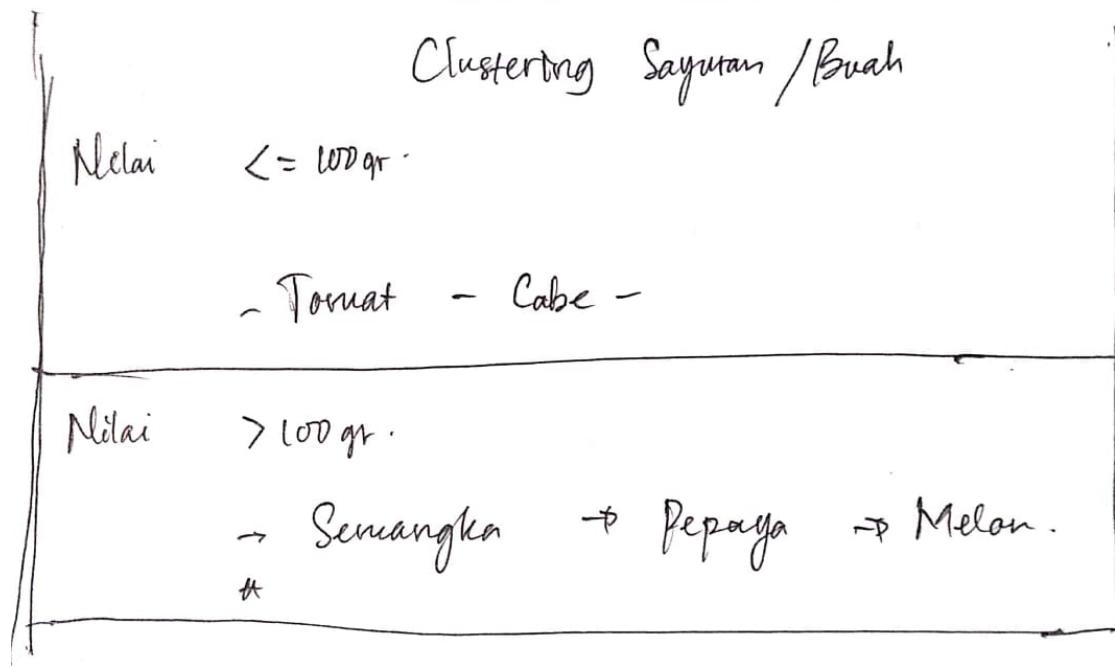


Figure 2.56: Ilustrasi Clustering

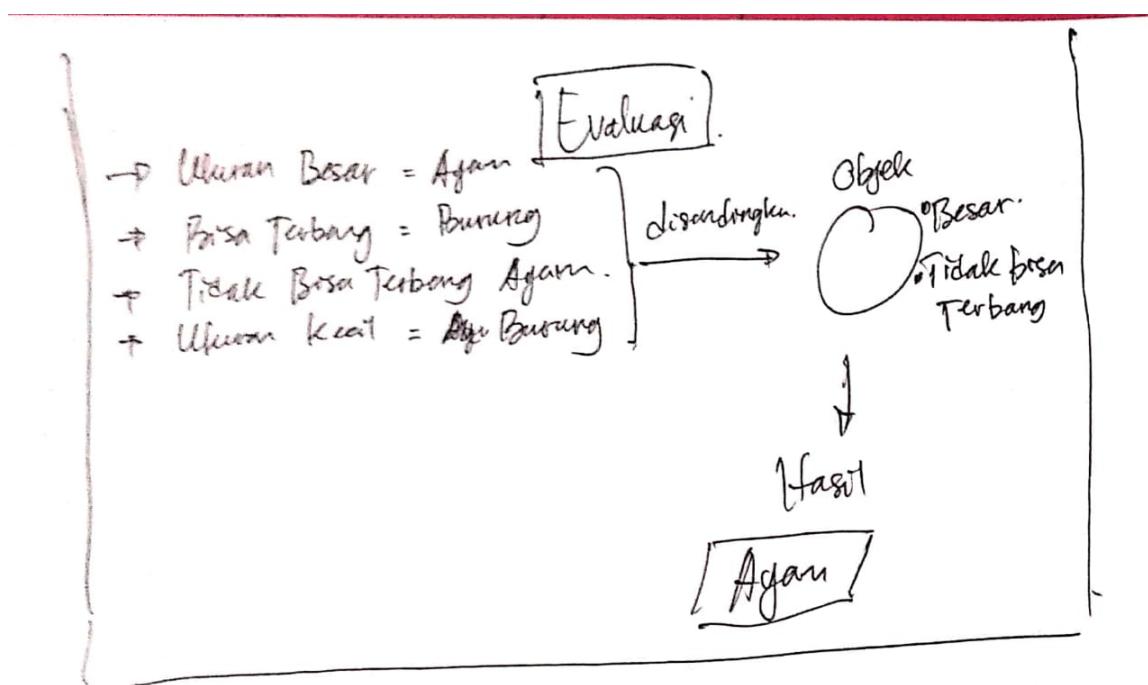


Figure 2.57: Ilustrasi Evaluasi

Confusion Matrix.

| | | | |
|---------|--------------|-------|---------|
| Kenanga | | | 30 |
| Mawar | | 80 | |
| Melati | 80 | | |
| | Bunga Melati | Mawar | Kenanga |

Confusion Matrix.

| | | | |
|---------|--------|-------|---------|
| Kenanga | 2 | 2 | 26 |
| Mawar | 2 | 27 | 1 |
| Melati | 25 | 3 | 2 |
| | Melati | Mawar | Kenanga |

Figure 2.58: Ilustrasi Confusion Matrix

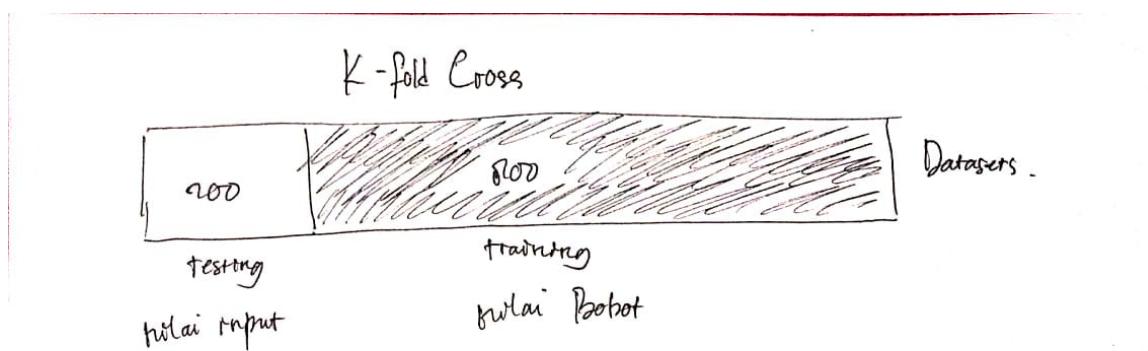


Figure 2.59: Ilustrasi K-Fold

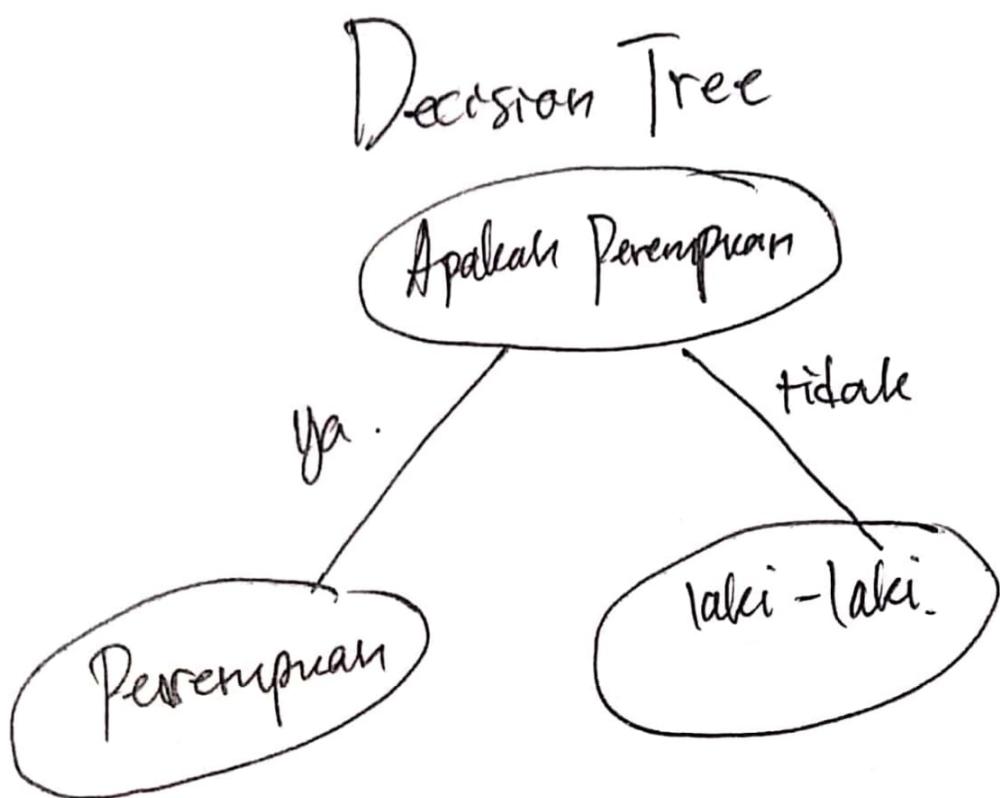
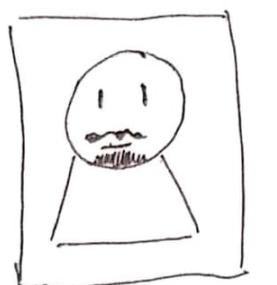


Figure 2.60: Ilustrasi Decision Tree

[Information Gain.]



Bewambut Pendek
memiliki jahut ✓
Berjenggot ✓
berkumur
bahu lebar ✓

60 % Akurat
40 % Tidak

Figure 2.61: Contoh Ilustrasi Information Gain.

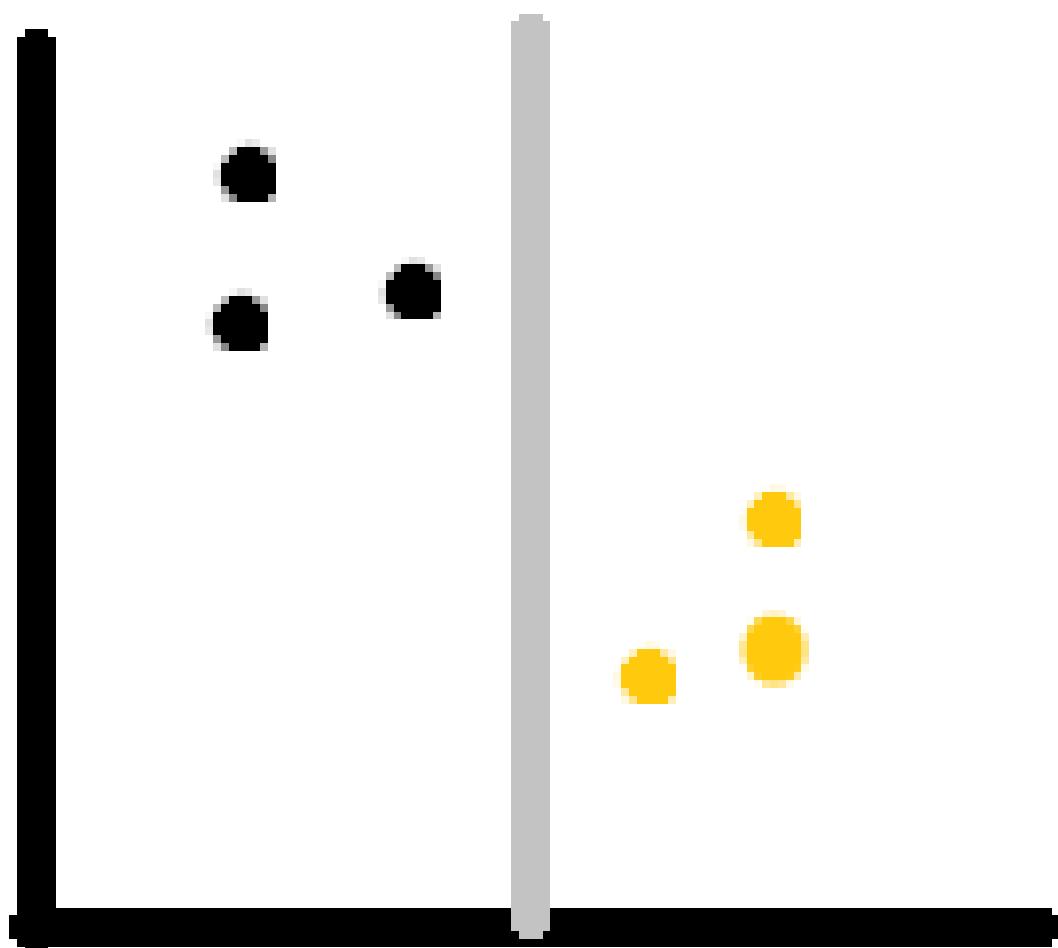


Figure 2.62: Contoh Penggunaan Binary Classification

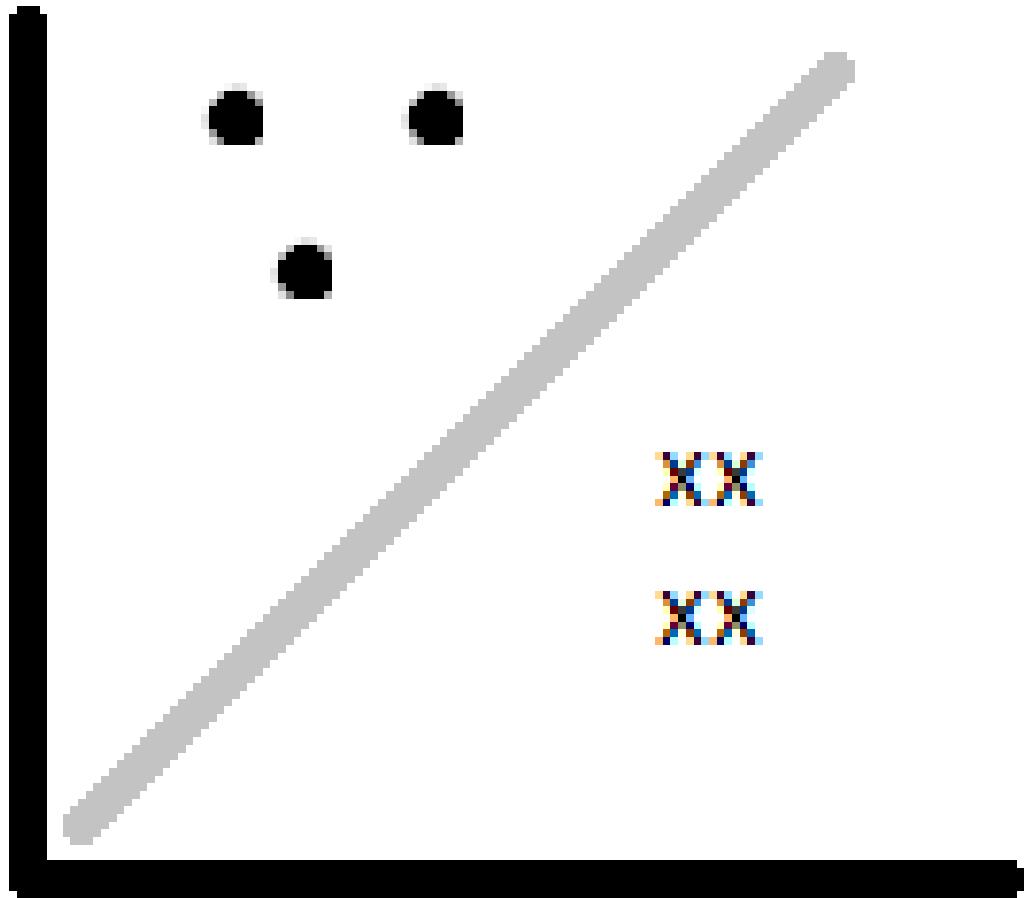
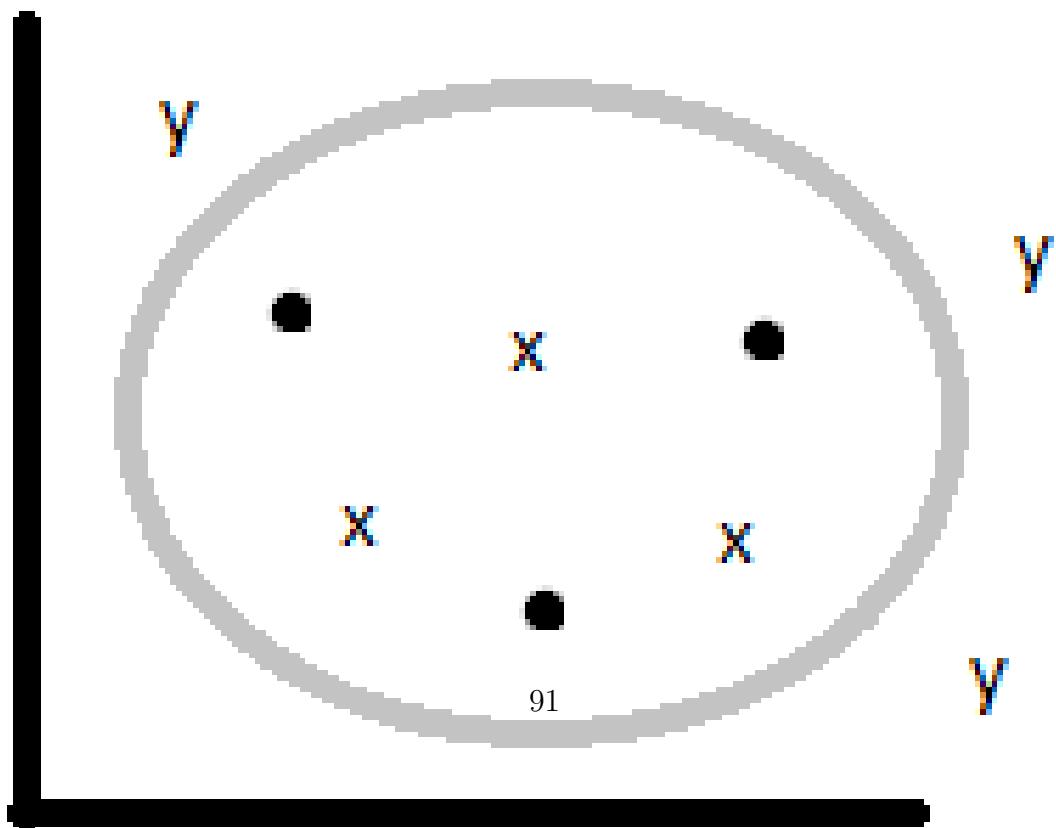


Figure 2.63: Contoh Penggunaan Supervised Learning



EVALUASI

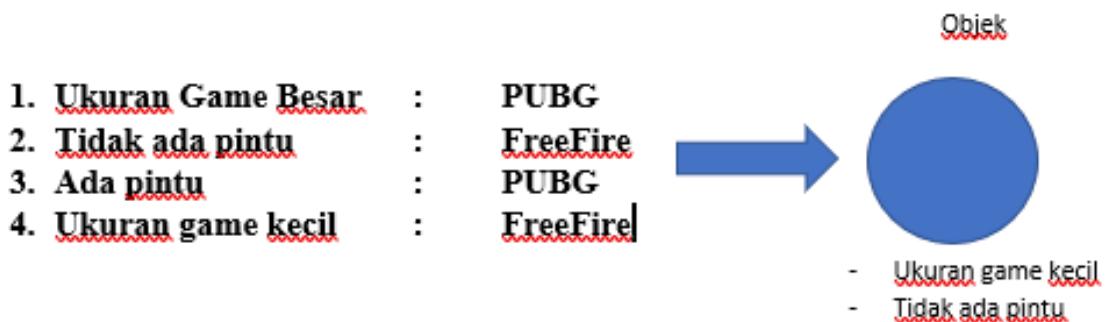


Figure 2.66: Contoh Penggunaan Evaluasi dan Akurasi

| | | | |
|------|----|----|----|
| 40 | | | 10 |
| 30 | | 10 | |
| 20 | 10 | | |
| usia | 20 | 30 | 40 |

Figure 2.67: Contoh Matrix Confusion

| | | | |
|------|----|----|----|
| 40 | 2 | 1 | 7 |
| 30 | 1 | 9 | 0 |
| 20 | 8 | 1 | 1 |
| usia | 20 | 30 | 40 |

Figure 2.68: Contoh Matrix Confusion

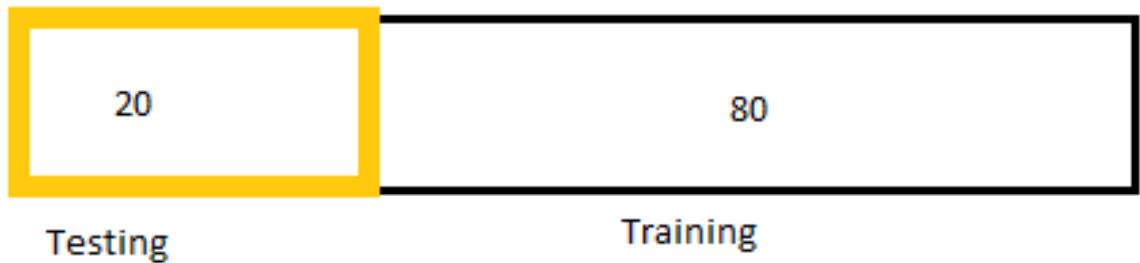


Figure 2.69: Contoh Penggunaan K Fold Cross Validation



Figure 2.70: Contoh Penggunaan Decision Tree



Figure 2.71: Contoh Penggunaan Information Gain

```
In [3]: import pandas as plum
....: durian = plum.read_csv('dataset/student-mat.csv', sep=';')
....: len(durian)
Out[3]: 395
```

Figure 2.72: code 1 hasil

```
In [5]: d['pass'] = d.apply(lambda row: 1 if (row['G1']+row['G2']+row['G3']) >= 35 else 0, axis=1)
....: d = d.drop(['G1', 'G2', 'G3'], axis=1)
....: d.head()
Out[5]:
   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 2.73: code 2 hasil

```
Out[5]:
   ...
   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 2.74: code 3 hasil

```

...: import numpy as npas
...: print("Passing: %d out of %d (%.2f%%)" % (npas.sum(d_pass),
len(d_pass), 100*float(npas.sum(d_pass)) / len(d_pass)))
Passing: 166 out of 395 (42.03%)

```

Figure 2.75: code 4 hasil

```

In [20]: from sklearn import tree
...: tomat = tree.DecisionTreeClassifier(criterion="entropy", max_depth=5)
...: tomat = tomat.fit(d_train_att, d_train_pass)

```

Figure 2.76: code 5 hasil

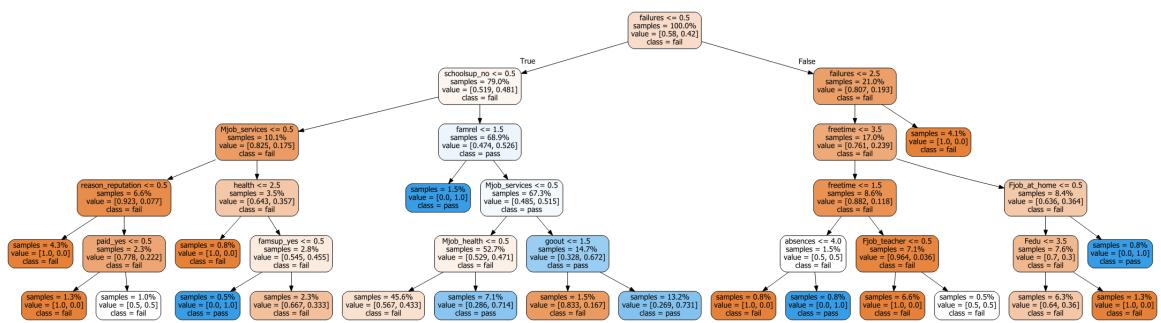


Figure 2.77: code 6 hasil

```

In [50]: tree.export_graphviz(tomat, out_file="student-performance.dot",
label="all", impurity=False, proportion=True,
...:                                     feature_names=list(d_train_att),
class_names=["fail", "pass"],
...:                                     filled=True, rounded=True)

```

Figure 2.78: code 7 hasil

Out[8]: 0.6644295302013423

Figure 2.79: code 8 hasil

Accuracy: 0.58 (+/- 0.09)

Figure 2.80: code 9 hasil

Max depth: 1, Accuracy: 0.58 (+/- 0.01)
Max depth: 2, Accuracy: 0.58 (+/- 0.03)
Max depth: 3, Accuracy: 0.57 (+/- 0.11)
Max depth: 4, Accuracy: 0.54 (+/- 0.06)
Max depth: 5, Accuracy: 0.59 (+/- 0.09)
Max depth: 6, Accuracy: 0.58 (+/- 0.10)
Max depth: 7, Accuracy: 0.57 (+/- 0.10)
Max depth: 8, Accuracy: 0.59 (+/- 0.11)
Max depth: 9, Accuracy: 0.60 (+/- 0.08)
Max depth: 10, Accuracy: 0.59 (+/- 0.03)
Max depth: 11, Accuracy: 0.58 (+/- 0.08)
Max depth: 12, Accuracy: 0.58 (+/- 0.10)
Max depth: 13, Accuracy: 0.57 (+/- 0.06)
Max depth: 14, Accuracy: 0.58 (+/- 0.10)
Max depth: 15, Accuracy: 0.58 (+/- 0.07)
Max depth: 16, Accuracy: 0.56 (+/- 0.07)
Max depth: 17, Accuracy: 0.57 (+/- 0.07)
Max depth: 18, Accuracy: 0.59 (+/- 0.06)
Max depth: 19, Accuracy: 0.58 (+/- 0.08)

Figure 2.81: code 10 hasil

```
array([[1.0000000e+00, 5.79751704e-01, 6.30768599e-03],  
       [2.0000000e+00, 5.84847452e-01, 2.64856147e-02],  
       [3.0000000e+00, 5.72122687e-01, 1.08161582e-01],  
       [4.0000000e+00, 5.41806232e-01, 5.92037007e-02],  
       [5.0000000e+00, 5.87280104e-01, 9.24946725e-02],  
       [6.0000000e+00, 5.87247647e-01, 9.98900892e-02],  
       [7.0000000e+00, 5.84684356e-01, 9.49754676e-02],  
       [8.0000000e+00, 5.87248458e-01, 1.04805546e-01],  
       [9.0000000e+00, 5.89782538e-01, 8.41669324e-02],  
       [1.0000000e+01, 5.94973223e-01, 5.83834054e-02],  
       [1.1000000e+01, 5.82154333e-01, 5.44767819e-02],  
       [1.2000000e+01, 5.72090230e-01, 8.18652096e-02],  
       [1.3000000e+01, 5.69623499e-01, 4.45760652e-02],  
       [1.4000000e+01, 5.59432814e-01, 6.41830829e-02],  
       [1.5000000e+01, 5.74750081e-01, 8.67458998e-02],  
       [1.6000000e+01, 5.51932814e-01, 7.98351552e-02],  
       [1.7000000e+01, 5.77250893e-01, 5.74806698e-02],  
       [1.8000000e+01, 5.69623499e-01, 4.16011588e-02],  
       [1.9000000e+01, 5.77345829e-01, 8.06083322e-02]])
```

Figure 2.82: code 11 hasil

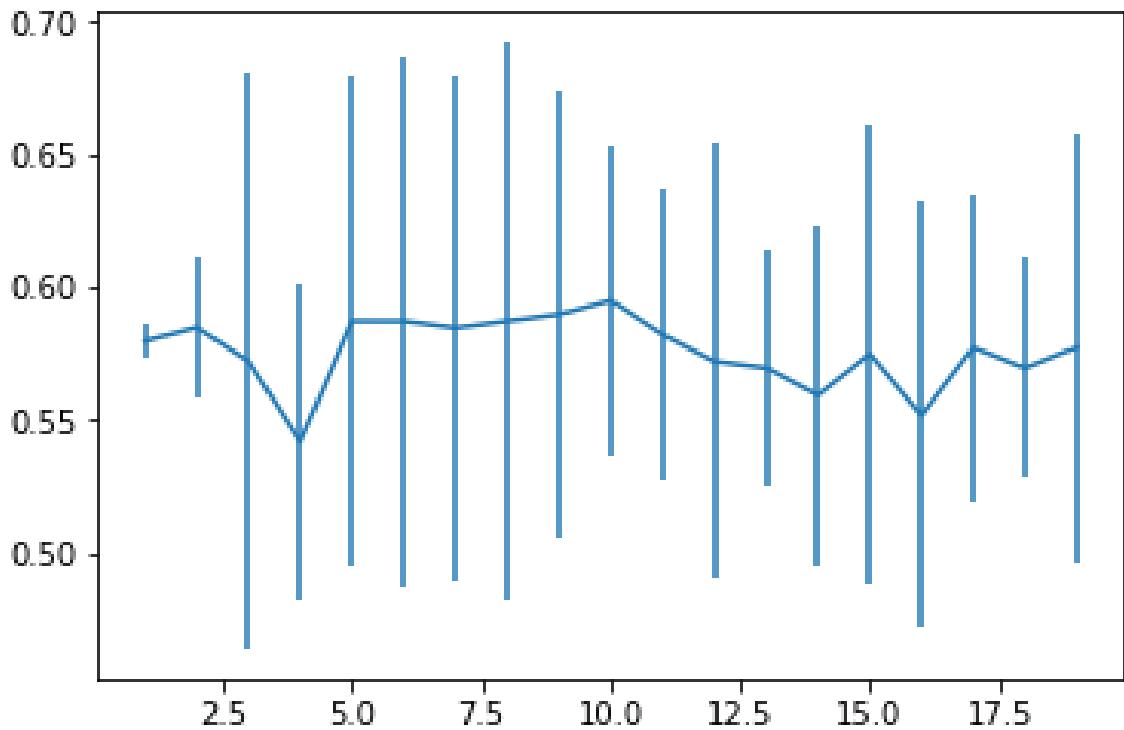


Figure 2.83: code 12 hasil

```
File "C:\Users\Fathi-PC\Anaconda3\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 2.84: Error Path

```
C:\Users\Fathi-PC>pip install graphviz
Collecting graphviz
  Using cached https://files.pythonhosted.org/
  /a/graphviz-0.10.1-py2.py3-none-any.whl
Installing collected packages: graphviz
Successfully installed graphviz-0.10.1

C:\Users\Fathi-PC>conda install graphviz
Collecting package metadata: done
Solving environment: done

*** Packages to install ***

```

Figure 2.85: Fix Error

Chapter 3

Methods

3.1 Fathi Rabbani / 1164074

3.1.1 Teori

1. Random Forest

Random forest adalah suatu algoritma yang digunakan pada klasifikasi data dalam jumlah yang besar. Klasifikasi random forest dilakukan melalui penggabungan pohon (tree) dengan melakukan training pada sampel data yang dimiliki. Penggunaan pohon (tree) yang semakin banyak akan mempengaruhi akurasi yang akan didapatkan menjadi lebih baik. Penentuan klasifikasi dengan random forest diambil berdasarkan hasil voting dari tree yang terbentuk. Pemenang dari tree yang terbentuk ditentukan dengan vote terbanyak. berikut adalah struktur dari Random Forest ada pada Gambar 6.10

2. Membaca Dataset, Makna setiap file dan Menjelaskan data CUB-200-2011

(a) Membaca Data

- dengan membuka data yang sudah didownload yaitu data CUB-200-2011 atau data tentang perbandingan data jenis burung,
- lalu data tersebut dibuka dengan menggunakan aplikasi Spyder, dan dijalankan setiap baris Code yang ada.
- data yang ada pada folder CUB-200-2011 dibuka dengan menggunakan code dari Chapter 2 yang ada pada buku pembelajaran.

(b) Makna setiap File

- data yang terdapat pada file CUB-200-2011 ada data folder ATTRIBUTE, IMAGES, PARTS yang memiliki kegunaannya sendiri yang dimana

pada penggunaannya data yang dipakai adalah data image_attribute_label pada folder attribute, data image_class_labels dan data classes.

- file image_attribute_label berguna sebagai data awal yang digunakan untuk membaca data attribute yang terdapat pada masing - masing gambar burung yang ada.
- sedangkan file image_class_label yang ada pada folder CUB-200-2011 berguna sebagai data yang akan membuat kolom baru pada dataset yang fungsinya adalah untuk memasukan hasil dari semua data yang dimiliki oleh imgatt2.
- dan file classes berguna sebagai dataset yang akan dipanggil oleh fungsi code untuk menampilkan nama dari data burung yang dimiliki.

(c) Isi Field

- file image_attribute_label berisi tentang data attribute yang ada pada data gambar file burung yang dimiliki difolder image pada CUB-200-2011
- file image_class_label berisi tentang data yang dimiliki oleh attribute dari image_attribute_label dimana data yang bernilai atau memiliki nilai disusun hingga menghasilkan data yang mudah dipahami.
- file classes berisi tentang data yang berguna untuk menampilkan data nama dari setiap data jenis burung yang dimiliki.

3. Cross Validation

Cross-validation adalah metode statistik yang dapat digunakan untuk mengevaluasi kinerja model atau algoritma dimana data dipisahkan menjadi dua subset yaitu data training dan data testing.

4. Score 44 Random Forest, 27 Decision Tree dan 29 SVM

- (a) merupakan hasil dari pengolahan tentang data jenis burung yang dimiliki setelah melalui proses pembagian data training dan testing yang menghasilkan score 44 persen sebagai pembanding bahwa data yang diolah tersebut bernilai 44 persen tingkat kebenarannya atau keakuratannya.
- (b) lalu pada penggunaan decision tree yang menghasilkan nilai 27 persen menjelaskan bahwa data yang diolah dengan menggunakan decision tree

sebagai fungsi pembandingannya itu lebih kecil tingkat keakuratan hasilnya yang dimana kita mencari keakuratan data dari setiap jenis burung yang ada.

- (c) sedangkan dengan menggunakan SVM menghasilkan nilai sebesar 29 persen yang dimana merupakan nilai nerta menjelaskan bahwa data yang diolah masih belum akurat tingkat kesamaannya dengan data jenis burung yang dimiliki.

dari penjelasan tersebut disimpulkan bahwa penggunaan random forest dalam menentukan keakuratan data itu lebih besar scorenya dibandingkan menggunakan decision tree dan SVM.

5. Membaca Confusion Matriks

```
import numpy as np
np.set_printoptions(precision=2)
plt.figure(figsize=(60,60), dpi=300)
plot_confusion_matrix(cm, classes=birds, normalize=True)
plt.show()
```

penggunaan code diatas adalah cara untuk membaca dataset jenis burung dengan metode confusion matriks. dimana contoh hasil dari membaca data dengan menggunakan confusion matriks adalah sebagai berikut : Gambar 6.11

6. Voting pada Random Forest voting pada random forest berguna untuk mengambil nilai yang akan digunakan sebagai bandingan dari masing - masing tree yang ada untuk menghasilkan nilai final sebagai data yang diinginkan, contohnya adalah seperti berikut ini : Gambar 6.12

3.1.2 Praktek

1. membaca data Hero Dota dengan Pandas

pada code ini akan dijelaskan maksud dan tujuan dari setiap baris code yang ada.

- pertama ada perintah import yang berguna untuk memanggil library pandas untuk digunakan aliaskan sebagai dota agar mudah dalam melakukan pemanggilan variable.

- lalu ada variable hero yang menampung data untuk ditampilkan.
 - lalu ada variable dh yang berguna sebagai variable yang akan memanggil proses dari data dota dengan menggunakan tipe DataFrame dengan data variable hero.
 - lalu perintah print untuk menampilkan hasilnya ada pada gambar 3.9.
2. membaca data Integer yang disusun dengan menggunakan Numpy
- pada code ini kita akan menggunakan numpy sebagai library dengan beberapa function codenya seperti eye(), reshape() dan dot(), langsung saja dijelaskan bagian dari masing - masing baris codenya.
- pertama ada import numpy as cat dengan menggunakan library numpy maka kita harus terlebih dahulu melakukan import librarynya agar dapat menggunakan function yang terdapat disana, lalu kita menggunakan aliasnya menjadi cat
 - lalu terdapat beberapa variable yang dibuat yaitu a, b dan c dengan memiliki data tersendiri pertama ada variable a dengan function eye() yang berguna untuk menghasilkan matrix identitas, lalu disikan dengan nilai 5, hasilnya adalah sebagai berikut ??.
 - lalu variable b dengan function arange dan reshape yang berguna untuk menyusun data menjadi 5 x 5 dengan jumlah data 25 hasilnya bisa dilihat pada gambar ??
 - dan terakhir ada data variable c dengan function dot yang berguna untuk memberikan nilai titik pada masing - masing datanya seperti pada gambar ??

3. membaca data variable dengan menggunakan Matplotlib

pada code ini kita menggunakan matplotlib sebagai librarynya untuk dapat membaca data yang akan kita buat, pada kali ini kita akan menggunakan matplotlib dengan parameter class pyplot.

- pertama kita akan memanggil data metplotlibnya dengan menggunakan perintah import matplotlib.pyplot dengan alias me.
- lalu pada code berikut terdapat 2 variable yaitu kecerdasan dan nilai untuk digunakan sebagai data acuan yang akan dibaca oleh function bar() dan akan ditampilkan dengan menggunakan function show() hasilnya adalah seperti pada gambar 3.8

4. Random Forest

hasil yang didapatkan dari menggunakan pemrosesan data Random Forest terdapat pada gambar 3.9, hasil dari keluarannya terdapat nilai 0.4553854276663147 yang jika kita masukan dalam proses ini menjadi 45 % dengan nilai yang didapat tersebut meyakinkan bahwa data yang diolah telah mencapai titik dimana data tersebut dapat dianalisa dengan tepat.

5. Confusion Matrix

hasil yang didapat dari proses menggunakan 3.10 confusion matrix adalah tampilnya gambar 3.20 dengan hasil berupa gambar tersebut dapat dipastikan bahwa nilai yang didapatkan memiliki beberapa titik yang dapat disusun seperti pada gambar berikut 3.21.

6. Decision tree dan SVM

- Decision Tree

hasil dari proses terdapat pada gambar 3.11 terdapat nilai 0.2713833157338965 yang diproses menjadi 27 % pada kasus ini, yang menjelaskan bahwa nilai tersebut merupakan nilai dari hasil analisa function decision tree terhadap data yang sudah disediakan.

- SVM

hasil dari proses terdapat pada gambar 3.12 terdapat nilai 0.28801478352692717 yang diproses menjadi 28 % pada kasus ini, yang menjelaskan bahwa nilai tersebut merupakan nilai dari hasil analisa menggunakan function SVM terhadap data yang sudah disediakan dan menghasilkan data tersebut.

7. Cross Validation

pada hasil dari proses cross validation ini terdapat 3 hasil dengan menggunakan function Random Forest, Decision Tree dan SVM, yang menghasilkan beberapa data yang terdapat pada gambar 3.13 sebagai data nilai Random Forest, 3.14 sebagai hasil data Decision tree dan 3.15 merupakan hasil dari SVM. dimana data pada Random Forest bernilai 0.44 , Decision Tree 0.26 dan SVM 0.26 yang berarti dari 3 function yang digunakan untuk melakukan analisa data penggunaan Random Forest lah yang menghasilkan nilai data paling signifikan sehingga data yang diolah dapat diketahui bahwasannya data tersebut sudah 44 % termasuk data valid atau benar.

8. Pengamatan pengamatan hasil data tersebut dibagi menjadi 2 yaitu dengan metode cross validation dan matplotlib, hasilnya adalah sebagai berikut ini :

- Cross Validation

data yang dihasilkan merupakan data pengamatan apakah nilai yang dikeluarkan merupakan nilai yang valid sehingga data yang dianalisa dapat diketahui kebenarannya dengan menggunakan nilai range 10 - 200 dengan lompatan 20 data dan dimulai dari 10 serta 5 perulangan data hasilnya sebanyak 50 kali, hasilnya terdapat pada gambar 3.16

- Matplotlib

data yang dihasilkan merupakan data pengamatan apakah nilai yang dikeluarkan merupakan nilai yang valid sehingga data yang dianalisa dapat diketahui kebenarannya dengan menggunakan class Axes3D yang akan menghasilkan data berupa gambar 3.16 dimana data yang terbaca adalah menggunakan nilai variable X Y Z dengan membandingkan nilai dari masing - masing data variable tersebut, hasilnya terdapat pada gambar 3.16

3.1.3 Error

1. Error pada gambar 4.31
2. Jenis Error adalah tidak terbacanya data karena kurangnya nilai yang dimasukan
3. Solusinya adalah dengan menambahkan nilai sesuai dengan jumlah masing - masing kolom hasilnya ada pada gambar 3.19

3.2 Cokro Edi Prawiro / 1164069

3.2.1 Teori

1. Jelaskan apa itu random forest, sertakan gambar ilustrasi buatan sendiri.

Random Forest atau hutan acak yaitu kumpulan dari pohon-pohon keputusan yang digunakan untuk membaca objek tertentu yang telah di sepakati untuk di baca dalam AI. pohon-pohon keputusan tersebut akan memunculkan hasil-hasil yang akan disimpulkan oleh random forest. pembagian jumlah data yang dimasukan kedalam decision tree pada random forest akan di bagi sama rata

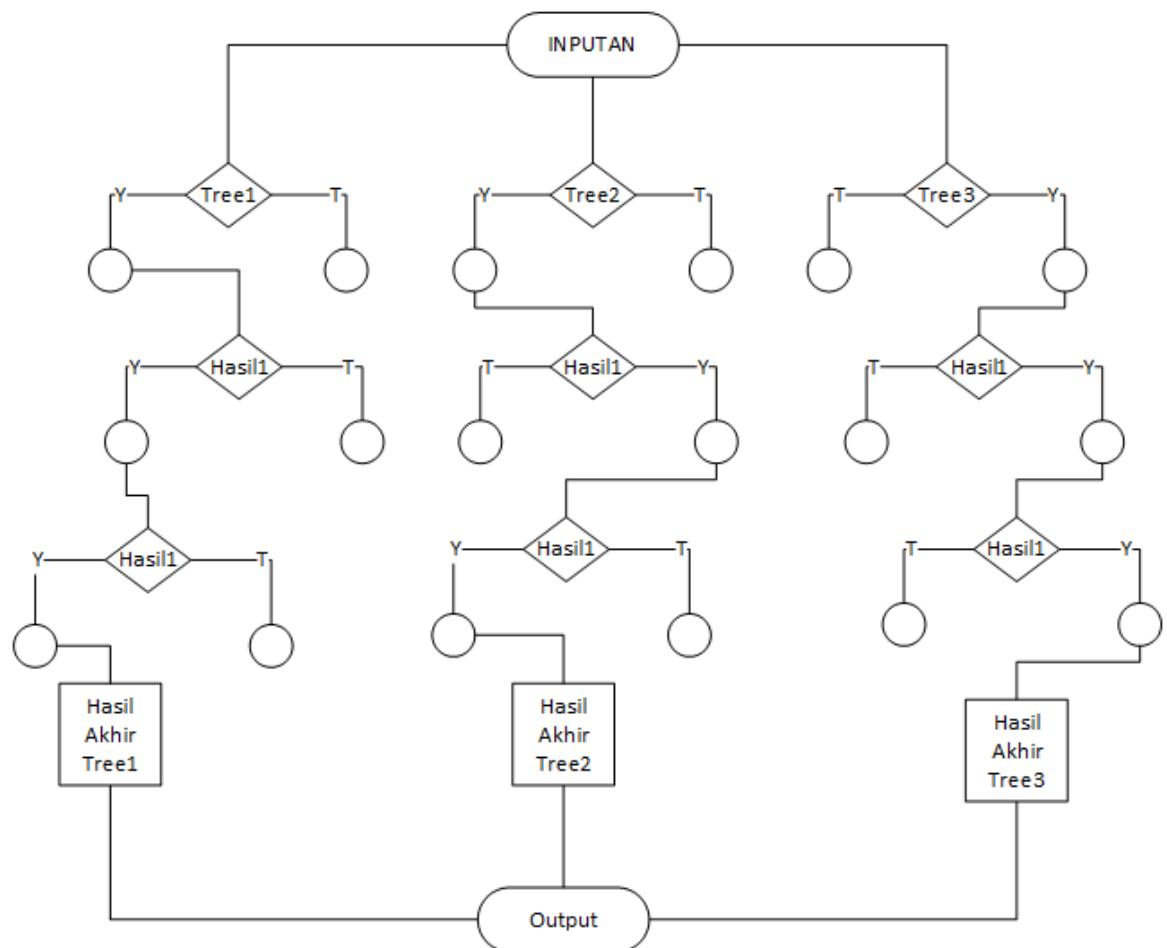


Figure 3.1: Random Forest

| | | Hasil Prediksi | |
|----------------|-------------------------|---------------------|---------------------|
| | | Bukan Gerakan | Gerakan |
| Hasil Validasi | Jumlah Seluruh Grid (N) | | |
| | Bukan Gerakan | True Negative (TN) | False Positive (FP) |
| | Gerakan | False Negative (FN) | True Positive (TP) |

Figure 3.2: Hasil dari membaca data dengan Confusion Matriks

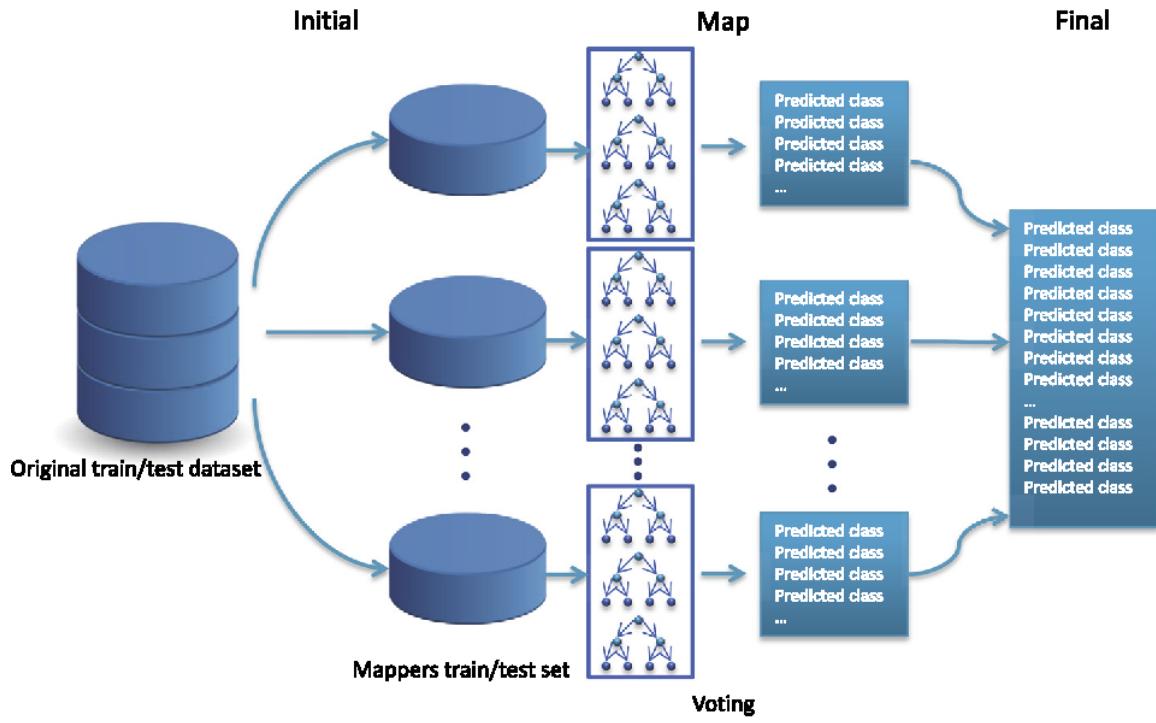


Figure 3.3: Voting Random Forest

```
In [47]: import pandas as dota
      ...: hero = {'Hero Name' : ['Juggernaut', 'Invoker', 'Axe', 'Crystal Maiden',
      ...: 'Pudge'],
      ...:         'Type Hero' : ['Agility', 'Intelegence', 'Strength', 'Support',
      ...: 'Tanker'],
      ...:         'Review 1-10' : [8,9,7,8,8]}
      ...: dh = dota.DataFrame(hero)
      ...: print(dh)
      Hero Name  Type Hero  Review 1-10
0   Juggernaut    Agility        8
1     Invoker  Intelegence        9
2       Axe      Strength        7
3  Crystal Maiden     Support        8
4       Pudge      Tanker        8
```

Figure 3.4: Pandas Implementasi

```
In [48]: a = cat.eye(5)
...: a
Out[48]:
array([[1.,  0.,  0.,  0.,  0.],
       [0.,  1.,  0.,  0.,  0.],
       [0.,  0.,  1.,  0.,  0.],
       [0.,  0.,  0.,  1.,  0.],
       [0.,  0.,  0.,  0.,  1.]])
```

Figure 3.5: Numpy Implementasi

```
In [49]: b = cat.arange(1,26).reshape(5,5)
...: b
Out[49]:
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

Figure 3.6: Numpy Implementasi

```

In [50]: c = cat.dot(a,b)
...: c
Out[50]:
array([[ 1.,  2.,  3.,  4.,  5.],
       [ 6.,  7.,  8.,  9., 10.],
       [11., 12., 13., 14., 15.],
       [16., 17., 18., 19., 20.],
       [21., 22., 23., 24., 25.]])

```

Figure 3.7: Numpy Implementasi

sesuai codingan atau ketentuan tertentu yang di sepakati. misalkan data yang akan digunakan sebanyak 314 jika dalam satu decision tree di putuskan untuk memiliki 50 data maka pada satu random forest akan terdapat enam atau tujuh decision tree. untuk lebih jelasnya dapat dilihat pemisalan pada gambar 3.22 random forest berikut.

2. Jelaskan cara membaca dataset khusus dan artikan makna setiap file dan isi field masing masing file. langkah pertama download terlebih dahulu dataset nya kemudian buka menggunakan spyder bawaan anaconda untuk mengetahui isi dari dataset tersebut. biasanya data tersebut berisi databerekstensi .txt yang di dalamnya terdapat class dari field atau data data yang ada data tersebut. contoh pada data burung ada field index dan angka, index biasanya berisi angka, angka angka tersebut memiliki makna yaitu pengganti nama atau jenis dari burung tersebut sedangkan pada field yang berisi nilai 0 dan 1 berarti menyatakan atau maknanya yaitu memberikan nilai ya dan tidak nilai tersebut di ubah menjadi angka nol dan satu karna data pada field tersebut harus berisi nilai boolean atau pilihan ya dan tidak di karenakan komputer susah membaca nilai dan tidak maka di ubahlah menjadi 0 dan 1 dengan 0 bernilai tidak dan 1 bernilai ya.
3. Jelaskan apa itu Cross Validation. Cross Validation merupakan cara untuk mengevaluasi hasil dari sebuah metode yang telah digunakan dengan cara mem-

```
In [46]: import matplotlib.pyplot as plt  
....:  
....: kecerdasan = ['Me', 'Friend']  
....: nilai = [65, 95]  
....:  
....: plt.bar(kecerdasan, nilai)  
....:  
....: plt.show()
```

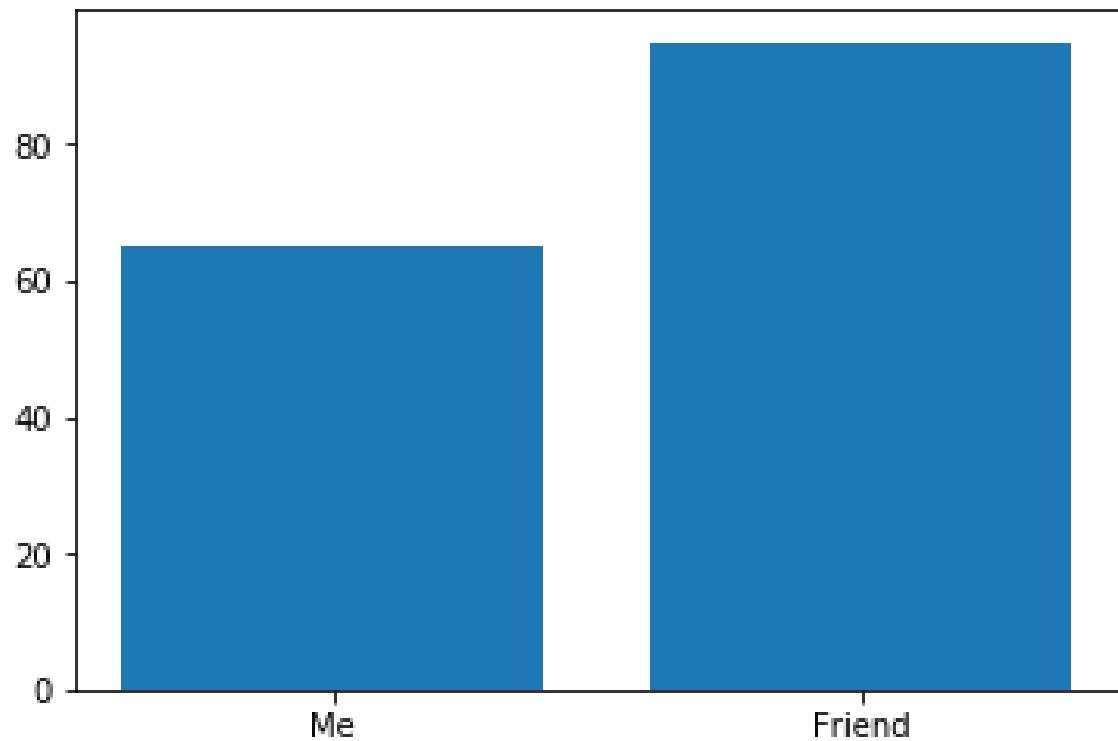


Figure 3.8: Matplotlib Implementasi

```

In [8]: clf.fit(df_train_att, df_train_label)
Out[8]:
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)

In [9]: print(clf.predict(df_train_att.head()))
[ 6 135  88 163 178]

In [10]: clf.score(df_test_att, df_test_label)
Out[10]: 0.4553854276663147

```

Figure 3.9: Random Forest Classifier

```

In [15]: import numpy as np
....: np.set_printoptions(precision=2)
....: plt.figure(figsize=(60,60), dpi=300)
....: plot_confusion_matrix(cm, classes=birds, normalize=True)
....: plt.show()
Normalized confusion matrix
[[0.15 0.05 0.2 ... 0. 0. 0.]
 [0. 0.55 0. ... 0. 0. 0.]
 [0. 0. 0.56 ... 0. 0. 0.]
 ...
 [0. 0. 0.05 ... 0.11 0. 0.]
 [0. 0. 0. ... 0.06 0.41 0.]
 [0. 0. 0. ... 0. 0. 0.81]]

```

Figure 3.10: Confusion Matrix

```

In [16]: from sklearn import tree
....: clftree = tree.DecisionTreeClassifier()
....: clftree.fit(df_train_att, df_train_label)
....: clftree.score(df_test_att, df_test_label)
Out[16]: 0.2713833157338965

```

Figure 3.11: Decision Tree

```
In [17]: from sklearn import svm
.... clfsvm = svm.SVC()
.... clfsvm.fit(df_train_att, df_train_label)
.... clfsvm.score(df_test_att, df_test_label)
C:\Users\Fathi-PC\Anaconda3\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[17]: 0.28801478352692717
```

Figure 3.12: Support Vector Machine

```
In [22]: 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))
Accuracy: 0.44 (+/- 0.03)
```

Figure 3.13: Cross Validation data Random Forest

```
In [23]: scorestree = cross_val_score(clftree, df_train_att, df_train_label, cv=5)
.... print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(), scorestree.std() * 2))
Accuracy: 0.26 (+/- 0.02)
```

Figure 3.14: Cross Validation data Decision Tree

```
In [21]: scoressvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv=5)
.... print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(), scoressvm.std() * 2))
C:\Users\Fathi-PC\Anaconda3\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)
C:\Users\Fathi-PC\Anaconda3\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)
C:\Users\Fathi-PC\Anaconda3\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)
C:\Users\Fathi-PC\Anaconda3\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)
C:\Users\Fathi-PC\Anaconda3\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)
C:\Users\Fathi-PC\Anaconda3\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: 0.26 (+/- 0.02)
```

Figure 3.15: Cross Validation data SVM

```

In [22]: max_features_opts = range(5, 50, 5)
....: n_estimators_opts = range(10, 200, 20)
....: 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: %.2f (+/- %.2f)" %
(max_features, n_estimators, scores.mean(), scores.std() * 2))
Max features: 5, num estimators: 10, accuracy: 0.26 (+/- 0.00)
Max features: 5, num estimators: 30, accuracy: 0.35 (+/- 0.02)
Max features: 5, num estimators: 50, accuracy: 0.38 (+/- 0.03)
Max features: 5, num estimators: 70, accuracy: 0.40 (+/- 0.03)

```

Figure 3.16: Pengamatan Akurasi data dengan Cross Validation

bagi dua bagian dari dataset menjadi data training dan data testing kemudian data tersebut diolah hingga muncul tingkat akurasi dari metode yang digunakan contoh pada metode random forest dataset nya di bagi menjadi dua menjadi data training dan data testing kemudian data tersebut di olah oleh mesin untuk melihat tingkat akurasinya maka akan muncul misalkan akurasi kebenaran sebesar 44 % begitu pula dengan menggunakan metode-metode yang lain seperti decision tree dan SVM.

4. Jelaskan apa arti score 44 % pada random forest, 27 % pada decision tree dan 29 % dari SVM. maksud dari score 44 % tersebut yaitu nilai ketepatan atau kebenaran atau bisa disebut hasil dari random forest misalkan dengan metode random forest mesin membaca objek burung, mesin tersebut bisa menyatakan jenis burung tersebut dengan akurasi kebenaran 44 %. sedangkan pada metode decision tree yaitu 27 % yang berarti menunjukkan bahwa tingkat akurasi ketepatan mesin jika mengerjakan sesuatu atau menyatakan keputusan dengan metode decision tree maka nilai kebenarannya bernilai 27 %. sedangkan dengan menggunakan metode SVM menunjukkan hasil 29 % yang berarti nilai ketepatan atau kebenaran dalam memecahkan masalah menggunakan metode SVM ini sebesar 29 % . maka dari itu dapat di simpulkan bahwa dengan menggunakan metode random forest mesin dapat memecahkan masalah lebih akurat dibandingkan dengan menggunakan decision tree dan SVM.
5. Jelaskan bagaimana cara membaca confusion matriks dan contohnya memakai

```
In [23]: import matplotlib.pyplot as plt
....: from mpl_toolkits.mplot3d import Axes3D
....: from matplotlib import cm
....: fig = plt.figure()
....: fig.clf()
....: 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(0.2, 0.5)
....: ax.set_xlabel('Max features')
....: ax.set_ylabel('Num estimators')
....: ax.set_zlabel('Avg accuracy')
....: plt.show()
```

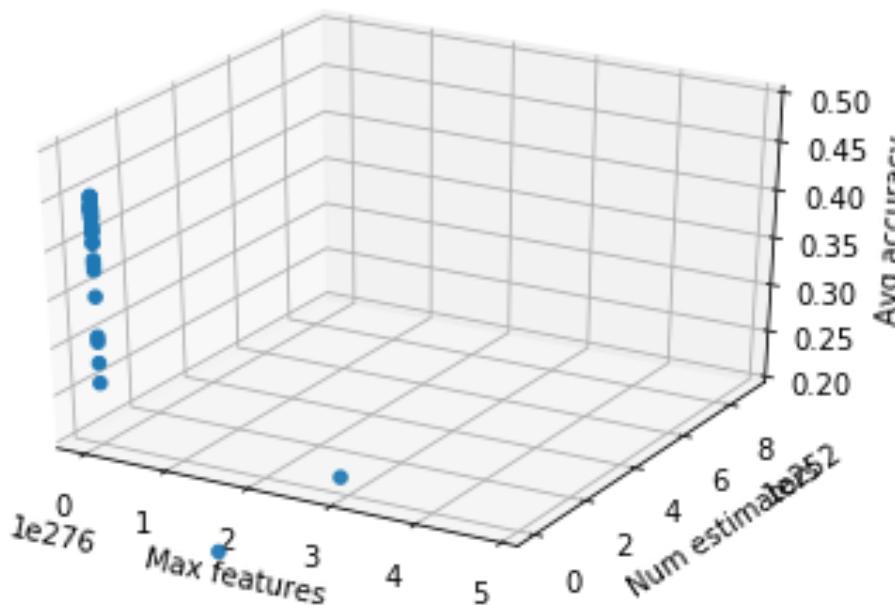


Figure 3.17: Pengamatan Akurasi data dengan Matplotlib

```
In [26]: import pandas as dota
.... hero = {'Hero Name' : ['Juggernaut', 'Invoker', 'Axe', 'Crystal Maiden', 'Pudge'],
....           'Type Hero' : ['Agility', 'Intelegence', 'Strength', 'Support'],
....           'Review 1-10' : [8,9,7,8]}
.... dh = dota.DataFrame(hero)
.... print(dh)
Traceback (most recent call last):

File "<ipython-input-26-a7c7a8250e1d>", line 5, in <module>
    dh = dota.DataFrame(hero)

File "C:\Users\Fathi-PC\Anaconda3\lib\site-packages\pandas\core\frame.py", line 348, in __init__
    mgr = self._init_dict(data, index, columns, dtype=dtype)

File "C:\Users\Fathi-PC\Anaconda3\lib\site-packages\pandas\core\frame.py", line 459, in _init_dict
    return _arrays_to_mngr(arrays, data_names, index, columns, dtype=dtype)

File "C:\Users\Fathi-PC\Anaconda3\lib\site-packages\pandas\core\frame.py", line 7356, in _arrays_to_mngr
    index = extract_index(arrays)

File "C:\Users\Fathi-PC\Anaconda3\lib\site-packages\pandas\core\frame.py", line 7402, in extract_index
    raise ValueError('arrays must all be same length')

ValueError: arrays must all be same length
```

Figure 3.18: Cross Validation data Random Forest

```
import pandas as dota
hero = {'Hero Name' : ['Juggernaut', 'Invoker', 'Axe', 'Crystal Maiden', 'Pudge'],
        'Type Hero' : ['Agility', 'Intelegence', 'Strength', 'Support', 'Tanker'],
        'Review 1-10' : [8,9,7,8,8]}
dh = dota.DataFrame(hero)
print(dh)
```

Figure 3.19: Cross Validation data Random Forest

| |
|------------------------------------|
| 180.Wilson_Warbler |
| 181.Worm_eating_Warbler |
| 182.Yellow_Warbler |
| 183.Northern_Waterthrush |
| 184.Louisiana_Waterthrush |
| 185.Bohemian_Waxwing |
| 186.Cedar_Waxwing |
| 187.American_Three_toed_Woodpecker |
| 188.Pileated_Woodpecker |
| 189.Red_bellied_Woodpecker |
| 190.Red_cockaded_Woodpecker |
| 191.Red_headed_Woodpecker |
| 192.Downy_Woodpecker |
| 193.Bewick_Wren |
| 194.Cactus_Wren |
| 195.Carolina_Wren |
| 196.House_Wren |
| 197.Marsh_Wren |
| 198.Rock_Wren |
| 199.Winter_Wren |
| 200.Common_Yellowthroat |
| 001.Black_faced_Albatross |
| 002.Laysan_Albatross |
| 003.Sooty_Albatross |
| 004.Groove_billed_Ani |
| 005.Crested_Auklet |
| 006.Least_Auklet |
| 007.Parakeet_Auklet |
| 008.Rhinoceros_Auklet |
| 009.Brewer_Blackbird |
| 010.Red_winged_Blackbird |
| 011.Rusty_Blackbird |
| 012.Yellow_headed_Blackbird |
| 013.Bullock |
| 014.Indigo_Bunting |
| 015.Lazuli_Bunting |
| 016.Painted_Bunting |
| 017.Cardinal |
| 018.Spectacled_Catbird |
| 019.Gray_Catbird |
| 020.Yellow_breasted_Chat |
| 021.Eastern_Towhee |
| 022.Chuck_will_Widow |
| 023.Brandt_Cormorant |
| 024.Red_faced_Cormorant |
| 025.Pelagic_Cormorant |

Figure 3.20: Hasil Confusion Matrix

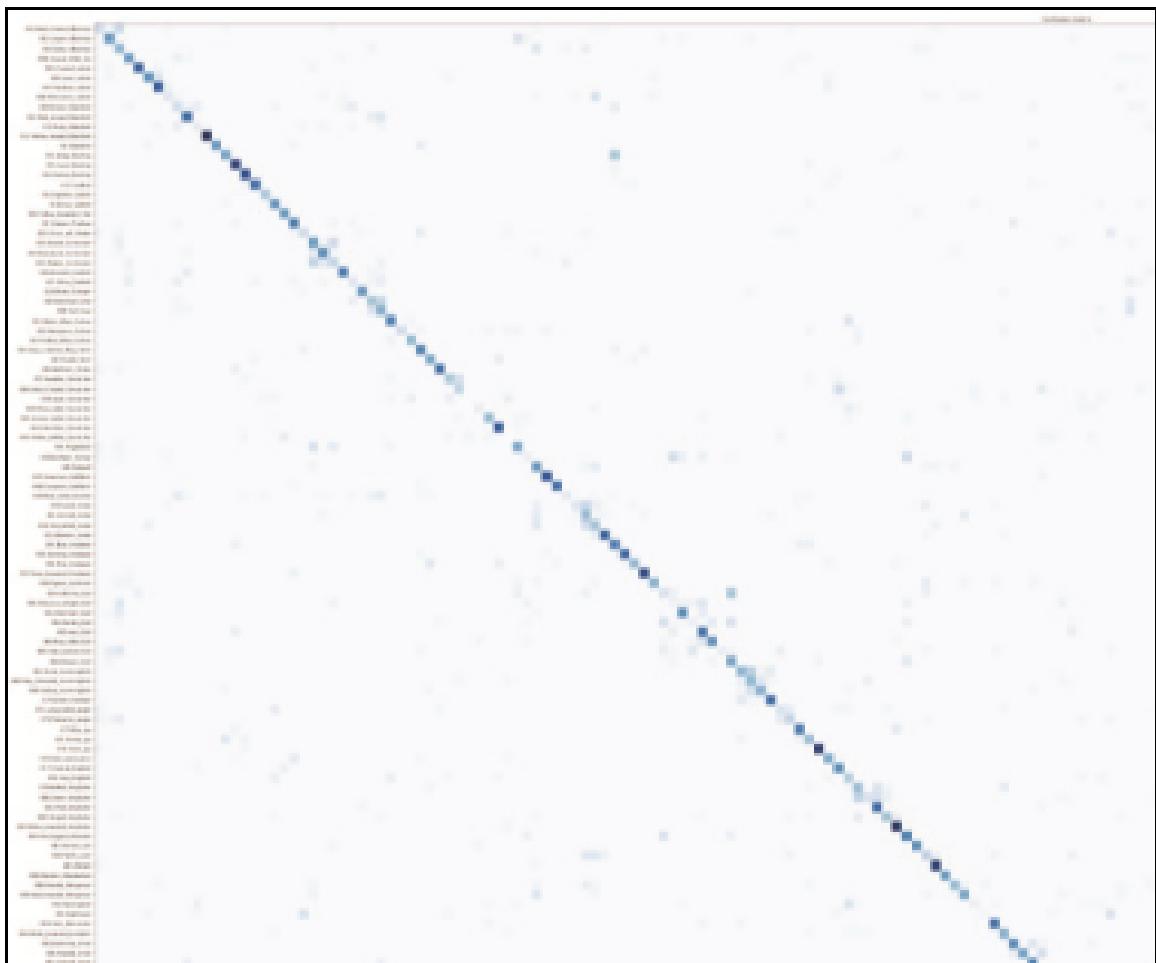


Figure 3.21: Hasil Confusion Matrix

gambar atau ilustrasi sendiri. cara membaca confusio matrix dengan cara memasukan para meter nilai yang ada pada datasets contoh pada dataset terdapat class yang disandingkan dengan nama burung untuk di normalisasi maka akan menunjukan nilai matrix yang mendekati nilai benar dalam bentuk angka misalkan 0,5 0,2 dan seterusnya mendekati nilai satu. di karenakan susahnya membaca nilai angka maka sering di ubah menjadi bentuk grafik. dapat di lihat pada gambar 3.24

6. Jelaskan apa itu voting pada random forest disertai dengan ilustrasi gambar sendiri.

Voting merupakan data hasil dari decision tree yang terdapat pada random forest. Dimana hasil data tersebut di gunakan sebagai acuan untuk hasil dari random forest. sebagai contoh misalkan pada satu random forest terdapat enam decision tree untuk menentukan jenis pekerjaan orang, pada decision tree ke satu menyimpulkan bahwa pekerjaanya yaitu dosen , pada decision tree ke dua yaitu dosen kemudian pada decision tiga dosen , pada decision tree ke empat yaitu pekerja kantoran, pada decision tree ke lima yaitu pekerja kantoran dan pada decision tree ke enam yaitu dosen. maka pada random forest dapat menyimpulkan hasilnya yaitu dosen. untuk lebih jelasnya dapat dilihat pada gambar 3.23 .

3.2.2 Praktikum

1. pandas

arti tiap baris codingan yang terdapat pada gambar 3.25 yaitu. pada baris ke satu yaitu perintah mengimport library padas pada python atau anaconda kemudian di inisialisasikan menjadi kue. selanjutnya pada baris ke 3 terdapat nama variabel yaitu nama_kue_tradisional = yang di dalamnya terdapat tiga nama field yakni Name Kue, harga satuan dan terbilang kemudian pada baris ke tujuh terdapat variabel baru bernama Data_kue = kemudian di dalamnya mendeskripsikan kue berdasarkan tipe DataFrame yang berisi variabel nama_kue_tradisional selanjutnya data tersebut di cetak pada console dengan perintah print (Data_kue). untuk lebih jelasnya dapat dilihat pada gambar 3.25. dan untuk hasilnya dapat dilihat pada gambar 3.26

2. numpy

Arti tiap baris codingan pada aplikasi sederhana numpy adalah sebagai berikut : pada baris ke satu yaitu mengimport numpy yang di inisialisasi menjadi np kemudian pada baris ke tiga dibut variabel ali yang berisi numpy bertipekan arrange 6 yang berarti berisi nilai array dari 0 sampai 5 kemudian pada baris ke empat di cetak hasilnya dengan memasukan perintah print (ali) selanjutnya yaitu membuat nilai array tiga dimensi pada baris ke tujuh dengan cara membuat variabel botak yang berisi rank nilainya kemudian dimensinya yaitu 4 3 3 kemudian variabel tersebut di print. selanjutnya pada baris ke 12 dibuat variabel nilai_array_1 dengan isian nilai array 1 2 3 4 kemudian pada baris ke 13 di buat variabe nilai_array_2 dengan nilai array 20 30 40 dan 50 selanjutnya pada baris ke 14 dibut nilai variabel Nilai_array_3 dengan rank 4 yang berarti berisi nilai dari 0 sampai 3 setelah itu di buat variabel hasil dimana isinya yaitu penjumlahan nilai_array_1+Nilai_array_3+Nilai_array_3 setelah itu nilai_array_1 , Nilai_array_3, dan Hasil di prin untuk melihat nilai dari array tersebut. untuk lebih jelasnya codingan aplikasi sederhana numpy dapat dilihat pada gambar 3.27 berikut:

3. matplotlib

Arti tiap baris aplikasi sederhana matplotlib pada baris ke satu yaitu memasukan library matplotlib.pyplot yang di definisikan menjadi plt kemudian membuat variabel kelas_ti3 pada baris ke tiga yang berisi label setelah itu di buat variabel jumlah_mhs3 pada baris ke empat yang berisi nilai dari setiap label tersebut. begitu juga pada baris ke emam dan ke tujuh kemudian pada baris ke sembilan matplotlib mendefinisikan gambar dengan ukurannya dan pada baris ke 10 di dekralasikan subplot setelah itu pada baris ke 11 matplotlib mendefinisikan jenis grafik yang digunakan dan dimasukan variabel kelas dan jumlah_mhs. begitujuga oada baris ke 13 14 dan 15 setelah itu di buat title pada baris ke 17 dan matplotlib di show untuk mendapatkan hasil dari grafiknya. untuk lebih jelasnya dapat di lihat pada gambar3.28 dan untuk hasilnya dapat dilihat pada gambar 3.29.

4. Random Forest

Arti tiap baris hasil codingan random forest pada baris pertama random forest di import dari sklearn dengan ketentuan yaitu maksimal isi dari decision tree berisi 50 data dengan keadaan random dan dengan estimators 100 data ini berada dalam variabel clf kemudian setelah itu variabel clf di running berdasarkan

data training dan data label yang telah di definisikan jumlahnya. kemudian variabel clf di running berdasarkan data training paling atas untuk memunculkan hasil data training lima paling atas. setelah itu data tersebut di running scorenya untuk melihat tingkat akurasi yang dia kerjakan maka tingkat akurasi yang dihasilkan berada pada kisaran 0,437 atau kisaran 43 % untuk lebih jelasnya dapat di lihat pada gambar 3.30 berikut.

5. Confusion Matrix

arti codingan pada hasil tiap codingan confusion matrix pada baris pertama codingan tersebut mendeskripsikan atau mengimport confusion matrix dari sklearn kemudian dibuat variabel pred labels dengan di isikan clf prdic df_test_att setelah itu membuat variabel cm yang isinya terdapat data yang di buat confusion matrix berdasarkan data test setelah variabel cm akan di running yang mana akan menghasilkan gambar berupa matrix matrix tersebut berisi nilai nilai kebenaran yang mendekati nilai benar atau mutlak nilai benar untuk hasilnya dapat di lihat pada gambar 3.31 data tersebut berisi data tipe int 64.

6. SVM dan Decision Tree

Arti dari setiap baris hasil codingan decision tree dan SVM pada tree masukan terlebih dahulu library tree setelah itu buat variabel clftree yang berisi decision tree setelah itu masukan nilai data training dan label yang telah di deklarasikan tadi setelah itu running variabel tersebut untuk mendapatkan score 0,266 kisaran 26 sampai 27 % akurasinya kemudian pada svm juga hampir sama masukan terlebih dahulu librarynya setelah itu buat variabel clfsvm yang berarti berisi nilai data training dan data label dan pendeklarasian svm itu sendiri setelah itu di running untuk mendapatkan nilai akurasinya atau score sebesar 0,283 atau dalam kisaran 23 %. untuk lebih jelasnya dapat di lihat pada gambar3.33 sebagai berikut:

7. Cross Validation

arti dari setiap baris hasil cross validation pada gambar3.34 tersebut diperlihatkan codingan error dikarenakan data training terlalu besar maka untuk mengatasinya dapat dilihat pada sub bab penanganan error / cokro

8. program pengamatan arti dari hasil program pengamatan. perogram pengamatan ini menggunakan library matplotlib supaya hasil dari presentase hasil random forest, svm dan decision tree dapat di bandingkan dengan membuat

variabel X Y Z kemudian memberikan label untuk setiap dimensinya untuk lebih jelas dapat dilihat gambar 3.35 yang menunjukan hasil perbandingan persentase dari tiga metode tersebut.

3.2.3 Penanganan Error / cokro

Screenshot error

1. Untuk gambar screenshot error yang pertama dapat dilihat pada gamabar 3.36
2. Untuk gambar screenshot error yang kedua dapat dilihat pada gamabar 3.37
3. Untuk gambar screenshot error yang ketiga dapat dilihat pada gamabar 3.38
4. Untuk gambar screenshot error yang keempat dapat dilihat pada gamabar 3.39

Code Errornya

1. kode error pada screenshot ke satu yaitu dikarenakan `clfsvm.fit(df_train_att, df_train_label)` dikarenakan data trainingnya terlalu besar sehingga komputernya error.
2. untuk kode error pada screen shoot ke 2 sampai ke 4 dikarenakan pada kode berikut `scores = cross_val_score(clf, df_train_att, df_train_label, cv=5)` `scorestree = cross_val_score(clftree, df_train_att, df_train_label, cv=5)` dan `scoressvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv=5)` hal ini di karenakan data trainingterlalu besar sehingga berdampak pada komputer sehingga library dari python tidak mampu mengolah data dan hasilnya menjadi error.

Solusi Untuk mengatasi Error

1. solusinya untuk yang ke satu yaitu dengan cara merestart spyder atau mematikannya kemudian nyalakan kembali setelah itu jalankan code yang error tersebut di CMD cika dalam python CMD jalam maka bisa di running. setelah itu buka kembali spyder dan jalankan codingan dari awal hingga pada bagian SVM tunggu sebenar sampai muncul nilai akurasinya.
2. solusi untuk mengatasi error tersebut yaitu dengan cara merubah bobot data pada data training seperti pada gambar 3.40 setelah itu running code yang errornya maka akan muncul hasilnya dengan skala yang lebih kecil dari persenan yang seharusnya. maka hasilnya seperti pada gambar 3.41 berikut.

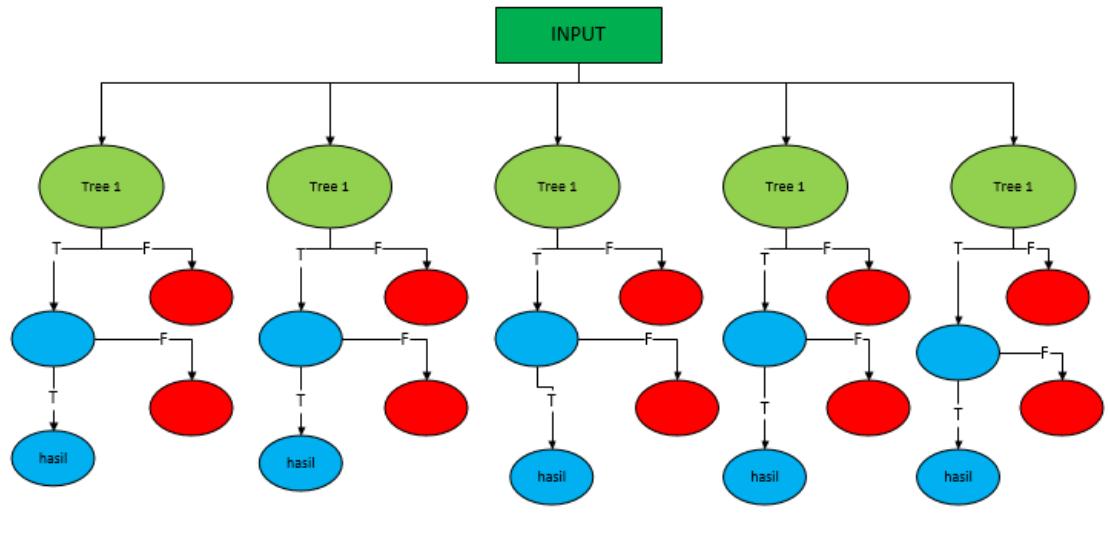


Figure 3.22: Random Forest

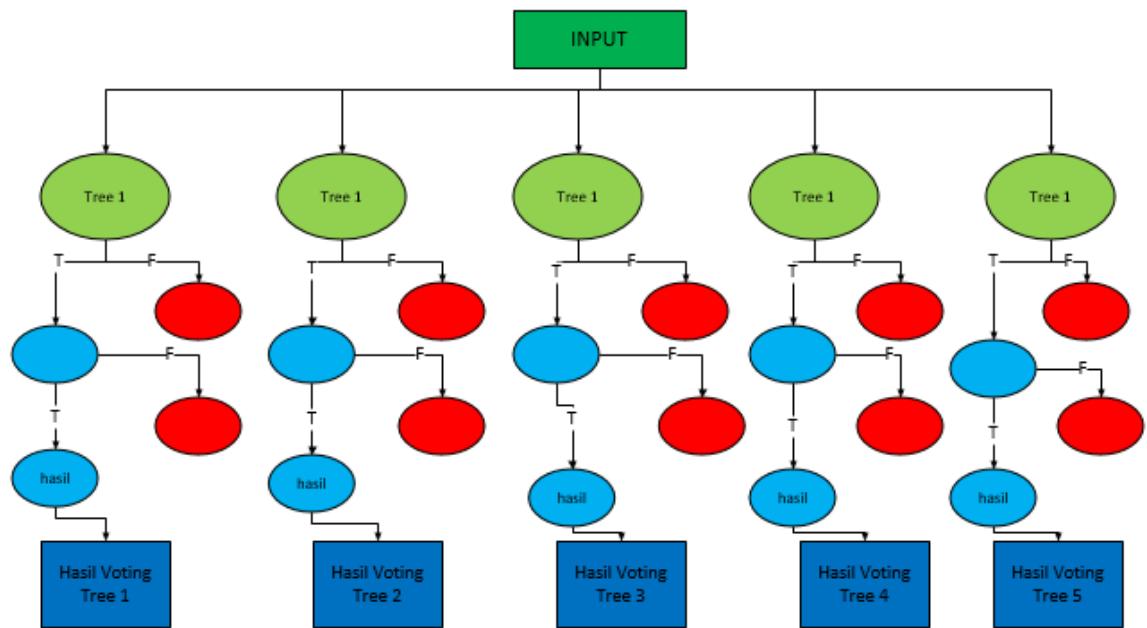


Figure 3.23: Ilustrasi Voting

| | | | | | | | |
|----------|----------|----------|----------|----------|----------|----------|----------|
| Burung a | | | | | | | |
| Burung b | | | | | | | |
| Burung c | | | | | | | |
| Burung d | | | | | | | |
| Burung e | | | | | | | |
| Burung f | | | | | | | |
| Burung g | | | | | | | |
| | Burung a | Burung b | Burung c | Burung d | Burung e | Burung f | Burung g |

Figure 3.24: Ilustrasi Confusion matrix

```

1 import pandas as kue
2
3 nama_kue_tradisional = {'Name Kue' : ['Putri Noong', 'Cuhcur', 'Bugis', 'Papais', 'Ali-ali'],
4 'Harga Satuan' : [2000, 5000, 1500, 2500, 1000],
5 'Terbilang' : ['Dua Ribu Rupiah', 'lima ribu rupiah',
6 'seribu limaratus rupiah', 'duaribu limaratus rupiah', 'seribu rupiah']}
7 Data_kue = kue.DataFrame(nama_kue_tradisional)
8 print (Data_kue)

```

Figure 3.25: Contoh aplikasi sederhana pandas

| Data_kue - DataFrame | | | |
|----------------------|-------------|--------------|----------------------------|
| Index | Name Kue | Harga Satuan | Terbilang |
| 0 | Putri Noong | 2000 | Dua Ribu Rupiah |
| 1 | Cuhcur | 5000 | lima ribu rupiah |
| 2 | Bugis | 1500 | seribu limaratus ru... |
| 3 | Papais | 2500 | duaribu limaratus ru... |
| 4 | Ali-ali | 1000 | seribu rupiah |

Figure 3.26: Hasil aplikasi sederhana pandas

```

1 import numpy as np
2
3 ali = np.arange(6)
4 print (ali)
5
6 #array 3dimensi
7 botak = np.arange(36).reshape(4,3,3)
8 print (botak)
9
10
11 #penjumlahan array
12 nilai_array_1 = np.array( [1,2,3,4] )
13 nilai_array_2 = np.array( [20,30,40,50] )
14 Nilai_array_3 = np.arange(4)
15 Hasil = nilai_array_1+Nilai_array_3+Nilai_array_3
16 print (nilai_array_1)
17 print (Nilai_array_3)
18 print (Hasil)

```

Figure 3.27: Contoh aplikasi sederhana numpy

```

1 import matplotlib.pyplot as plt
2
3 kelas_ti3 = ['Kelas A', 'Kelas B', 'Kelas C']
4 jumlah_mhs3 = [24, 27, 17]
5
6 kelas_ti2 = ['Kelas A', 'Kelas B', 'Kelas C']
7 jumlah_mhs2 = [22, 15, 24]
8
9 plt.figure(1, figsize=(9, 3))
10 plt.subplot(131)
11 plt.bar(kelas_ti3, jumlah_mhs3)
12
13 plt.figure(2, figsize=(9, 3))
14 plt.subplot(132)
15 plt.bar(kelas_ti2, jumlah_mhs2)
16
17 plt.suptitle('Categorical Plotting')
18 plt.show()

```

Figure 3.28: Contoh aplikasi sederhana matplotlib

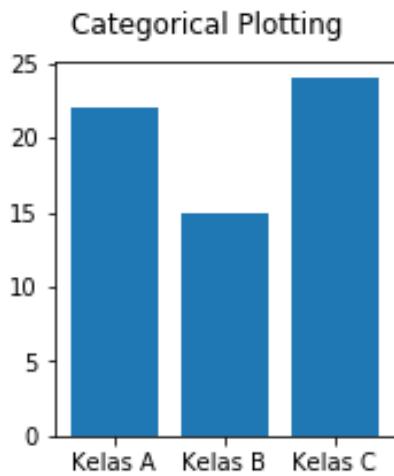
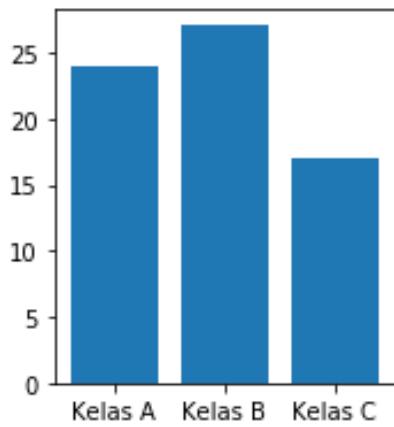


Figure 3.29: Hasil aplikasi sederhana matplotlib

```
In [60]: from sklearn.ensemble import RandomForestClassifier
...: clf = RandomForestClassifier(max_features=50, random_state=0, n_estimators=100)

In [61]: clf.fit(df_train_att, df_train_label)
Out[61]:
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)

In [62]: print(clf.predict(df_train_att.head()))
[ 4 81 73 34 14]

In [63]: clf.score(df_test_att, df_test_label)
Out[63]: 0.43769799366420276
```

Figure 3.30: Hasil aplikasi random forest

```
In [64]: from sklearn.metrics import confusion_matrix
....: pred_labels = clf.predict(df_test_att)
....: cm = confusion_matrix(df_test_label, pred_labels)

In [65]: cm
Out[65]:
array([[ 5,  2,  3, ...,  0,  0,  0],
       [ 0, 15,  0, ...,  0,  1,  0],
       [ 1,  0, 12, ...,  0,  0,  0],
       ...,
       [ 0,  0,  1, ...,  6,  0,  0],
       [ 0,  0,  0, ...,  0,  5,  0],
       [ 0,  0,  0, ...,  0,  0, 17]], dtype=int64)
```

Figure 3.31: Hasil aplikasi confusion matrix

```
In [24]: from sklearn import tree
....: clftree = tree.DecisionTreeClassifier()
....: clftree.fit(df_train_att, df_train_label)
....: clftree.score(df_test_att, df_test_label)
Out[24]: 0.26689545934530096

In [25]: from sklearn import svm
....: clfsvm = svm.SVC()
....: clfsvm.fit(df_train_att, df_train_label)
....: clfsvm.score(df_test_att, df_test_label)
C:\ProgramData\Anaconda3\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[25]: 0.28352692713833155
```

Figure 3.32: Hasil aplikasi SVM dan Decision tree

```
In [24]: from sklearn import tree
....: clftree = tree.DecisionTreeClassifier()
....: clftree.fit(df_train_att, df_train_label)
....: clftree.score(df_test_att, df_test_label)
Out[24]: 0.26689545934530096

In [25]: from sklearn import svm
....: clfsvm = svm.SVC()
....: clfsvm.fit(df_train_att, df_train_label)
....: clfsvm.score(df_test_att, df_test_label)
C:\ProgramData\Anaconda3\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[25]: 0.28352692713833155
```

Figure 3.33: Hasil aplikasi SVM dan Decision tree

```
In [41]: 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: %.2f (+/- %.2f)" % (scores.mean(), scores.std() * 2))
Traceback (most recent call last):

File "<ipython-input-41-42a1fd117fe3>", line 2, in <module>
    scores = cross_val_score(clf, df_train_att, df_train_label, cv=5)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 402, in cross_val_score
    error_score=error_score)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 240, in cross_validate
    for train, test in cv.split(X, y, groups))

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 917, in __call__
    if self.dispatch_one_batch(iterator):

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 759, in dispatch_one_batch
    self._dispatch(tasks)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 716, in _dispatch
    job = self._backend.apply_async(batch, callback=cb)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\_parallel_backends.py", line 182, in apply_async
    result = ImmediateResult(func)

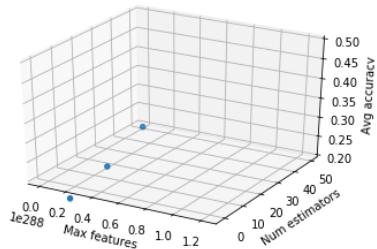
File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\_parallel_backends.py", line 549, in __init__
    self.results = batch()

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 225, in __call__
    for func, args, kwargs in self.items]

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 225, in <listcomp>
```

Figure 3.34: Hasil aplikasi cross validation

```
In [60]: import matplotlib.pyplot as plt
...: from mpl_toolkits.mplot3d import Axes3D
...: from matplotlib import cm
...: fig = plt.figure()
...: fig.clf()
...: ax = fig.gca(projection='3d')
...: x = rf_params[:,0]
...: y = rf_params[:,1]
...: z = rf_params[:,2]
...: ax.scatter(x, y, z)
...: ax.set_xlim(0,2,0.5)
...: ax.set_xlabel('Max features')
...: ax.set_ylabel('Num estimators')
...: ax.set_zlabel('Avg accuracy')
...: plt.show()
```



```
In [61]:
```

Figure 3.35: Hasil aplikasi cross validation

```

In [77]: from sklearn import svm
...: clfsvm = svm.SVC()
...: clfsvm.fit(df_train_att, df_train_label)
...: clfsvm.score(df_test_att, df_test_label)
Traceback (most recent call last):

File "<ipython-input-77-4515dd9b8c06>", line 3, in <module>
    clfsvm.fit(df_train_att, df_train_label)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py", line 178, in fit
    X_std = X.std()

File "C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\_methods.py", line 135, in _std
    keepdims=keepdims)

File "C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\_methods.py", line 112, in _var
    x = asanyarray(arr - arrmean)

MemoryError

```

Figure 3.36: Error Svm

```

In [46]: 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))
Traceback (most recent call last):

File "<ipython-input-46-42a1fd117fe3>", line 2, in <module>
    scores = cross_val_score(clf, df_train_att, df_train_label, cv=5)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 402, in cross_val_score
    error_score=error_score)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 240, in cross_validate
    for train, test in cv.split(X, y, groups))

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 917, in __call__
    if self.dispatch_one_batch(iterator):

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 759, in dispatch_one_batch
    self._dispatch(tasks)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 716, in _dispatch
    job = self._backend.apply_async(batch, callback=cb)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\_parallel_backends.py", line 182, in apply_async
    result = ImmediateResult(func)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\_parallel_backends.py", line 549, in __init__
    self.results = batch()

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 225, in __call__
    for func, args, kwargs in self.items]

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 225, in <listcomp>

```

Figure 3.37: Error Svm

```

IPython console
Console 1/A

In [47]:
In [47]: scorestree = cross_val_score(clftree, df_train_att, df_train_label, cv=5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(), scorestree.std() * 2))
Traceback (most recent call last):

File "<ipython-input-47-348c5d54f90>", line 1, in <module>
    scorestree = cross_val_score(clftree, df_train_att, df_train_label, cv=5)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 402, in cross_val_score
    error_score=error_score)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 240, in cross_validate
    for train, test in cv.split(X, y, groups))

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 917, in __call__
    if self.dispatch_one_batch(iterator):

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 759, in dispatch_one_batch
    self._dispatch(tasks)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 716, in _dispatch
    job = self._backend.apply_async(batch, callback=cb)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\_parallel_backends.py", line 182, in apply_async
    result = ImmediateResult(func)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\_parallel_backends.py", line 549, in __init__
    self.results = batch()

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 225, in __call__
    for func, args, kwargs in self.items]

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 225, in <listcomp>

```

Figure 3.38: Error Svm

```

In [48]:
In [48]: scoressvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv=5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(), scoressvm.std() * 2))
Traceback (most recent call last):

File "<ipython-input-48-d7a7153ce4e9>", line 1, in <module>
    scoressvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv=5)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 402, in cross_val_score
    error_score=error_score)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 240, in cross_validate
    for train, test in cv.split(X, y, groups))

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 917, in __call__
    if self.dispatch_one_batch(iterator):

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 759, in dispatch_one_batch
    self._dispatch(tasks)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 716, in _dispatch
    job = self._backend.apply_async(batch, callback=cb)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\_parallel_backends.py", line 182, in apply_async
    result = ImmediateResult(func)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\_parallel_backends.py", line 549, in __init__
    self.results = batch()

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 225, in __call__
    for func, args, kwargs in self.items]

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 225, in <listcomp>

```

Figure 3.39: Error Svm

```
In [53]: df_train_att = df_att[:600]
....: df_train_label = df_label[:600]
....: df_test_att = df_att[600:]
....: df_test_label = df_label[600:]
....:
....: df_train_label = df_train_label['label']
....: df_test_label = df_test_label['label']
```

Figure 3.40: Merubah data training

```
In [56]: 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))
C:\ProgramData\Anaconda3\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: 0.14 (+/- 0.08)

In [57]: scorestree = cross_val_score(clftree, df_train_att, df_train_label, cv=5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(), scorestree.std() * 2))
C:\ProgramData\Anaconda3\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: 0.12 (+/- 0.07)
```

Figure 3.41: Hasil Solusi

3.3 Ahmad Syafrizal Huda / 1164062

3.3.1 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 random forest pada gambar 3.42.
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 3.43.

Penjelasan dari isi field pada hasil gambar 3.43 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 kedapalm 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 outputanya untuk hutan acak, arti score 27% pada decission tree adalah presentasi hasil dari perhitungan dataset acak, dan arti score 29% dari SVM adalah hasil pendekatan jaringan saraf. Hasil tersebut didapat dari hasil valdasi 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 presen-tase 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.Greens):
    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 3.44 :

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 3.45.

3.3.2 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 3.46.

```
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 3.47.

```
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 insiasi 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 3.48.

4. Menjalankan Klasifikasi Random Forest.

Pada gambar 3.49 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 eror pada saat pembacaan file.

Pada gambar 3.50 yaitu mengembalikan baris n teratas (5 secara default) dari dataframe imgatt.

Pada gambar 3.51 yaitu menampilkan beberapa baris dan kolom dari dataframe imgatt.

Pada gambar 3.52 yaitu variabel imgatt2 menggunakan function pivot untuk mengubah kolom jadi baris, dan baris jadi kolom dari dataframe sebelumnya.

Pada gambar 3.53 yaitu imgatt2 head berfungsi untuk mengembalikan nilai teratas dari dataframe imgatt2.

Pada gambar 3.54 yaitu menampilkan beberapa baris dan kolom dari dataframe imgatt2.

Pada gambar 3.55 yaitu melakukan pivot yang mana imgid menjadi index yang artinya unik.

Pada gambar 3.56 yaitu memuat jawabannya yang berisi apakah burung itu termasuk dalam spesies yang mana. Dua kolomnya terdiri dari imgid dan label.

Pada gambar 3.57 yaitu menunjukkan 11788 baris dan 1 kolom. Dimana kolom itu adalah jenis spesies burungnya.

Pada gambar 3.58 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 3.59 yaitu menghilangkan label yang didepan, dan menggunakan label yang paling belakang yang baru di join.

Pada gambar 3.60 yaitu mengecek 5 data teratas dari df att.

Pada gambar 3.61 yaitu mengecek 5 data teratas dari df label.

Pada gambar 3.62 yaitu 8000 row pertama sebagai data training sisanya sebagai data testing dengan membagi menjadi dua bagian.

Pada gambar 3.63 yaitu memanggil kelas RandomForestClassifier. max features diartikan sebagai berapa banyak kolom pada setiap tree dengan isi maksimum 50.

Pada gambar 3.64 yaitu melakukan fit untuk membangun random forest yang sudah ditentukan dengan maksimum fitur sebanya 50 untuk perpohonnya.

Pada gambar 3.65 yaitu menampilkan hasil prediksi dari random forest sebelumnya.

Pada gambar 3.66 yaitu menampilkan besaran akurasinya dari prediksi di atas atau score perolehan dari klasifikasi.

5. Menjalankan Confusion Matrix.

Pada gambar 3.67 yaitu menyatukan Random Forest ke dalam Confusion Matrix

Pada gambar 3.68 yaitu menampilkan hasil dari gambar sebekumnya dengan array pada perintah cm.

Pada gambar 3.69 yaitu Plotting Confusion Matrix dengan librari Matplotlib.

Pada gambar 3.70 yaitu Set plot sumbunya sesuai dengan nama datanya dan membaca file classes.txt.

Pada gambar 3.71 yaitu Plot hasil perubahan label yang sudah dilakukan.

6. Menjalankan Klasifikasi SVM dan Decision Tree.

Pada gambar 3.72 yaitu klasifikasi dengan decission tree dengan dataset yang sama dan akan muncul akurasi prediksinya.

Pada gambar 3.73 yaitu klasifikasi dengan SVM dengan dataset yang sama dan akan muncul akurasi prediksinya.

7. Menjalankan Cross Validation.

Pada gambar 3.74 yaitu Cross Validation untuk Random Forest.

Pada gambar 3.75 yaitu Cross Validation untuk Decission Tree.

Pada gambar 3.76 yaitu Cross Validation untuk SVM.

8. Menjalankan Pengamatan Komponen Informasi.

Pada gambar 3.77 yaitu untuk mengetahui berapa banyak tree yang dibuat, berapa banyak atribut yang dipakai dan informasi lainnya.

Pada gambar 3.78 yaitu hasil dari plotting komponen informasi agar dapat dibaca.

3.3.3 Penanganan Eror

1. Screenshootan eror ada pada gambar 3.79.

```
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)" % (max_features, n_estimators, scores.mean(), scores.std() * 2))

```

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

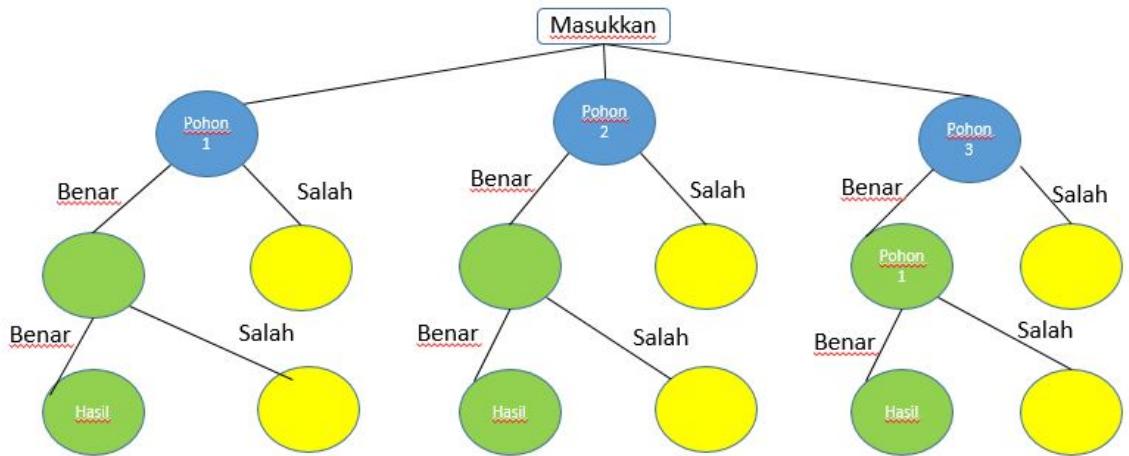


Figure 3.42: Random Forest

| 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 3.43: Hasil Dataset

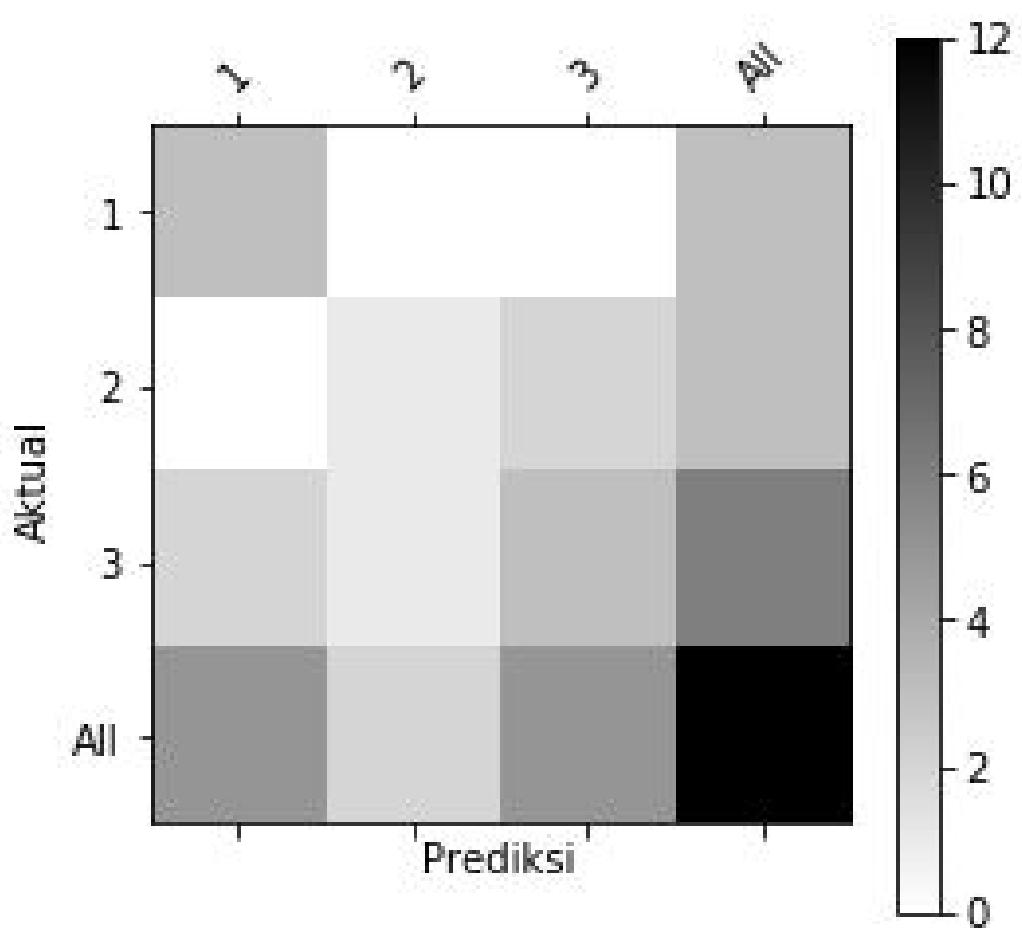


Figure 3.44: Confusion Matrix

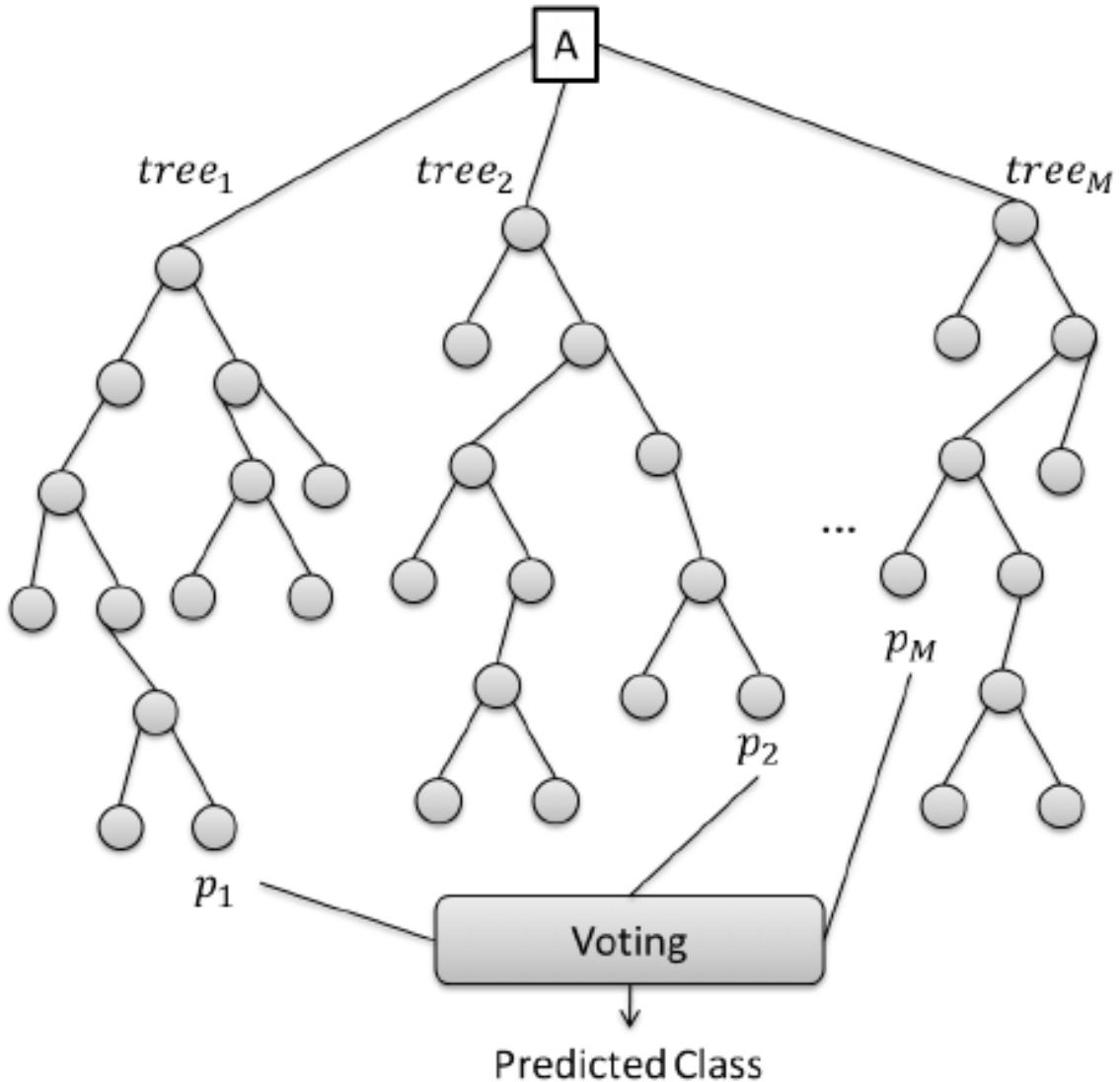


Figure 3.45: Voting

```
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 3.46: 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 3.47: 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()
```

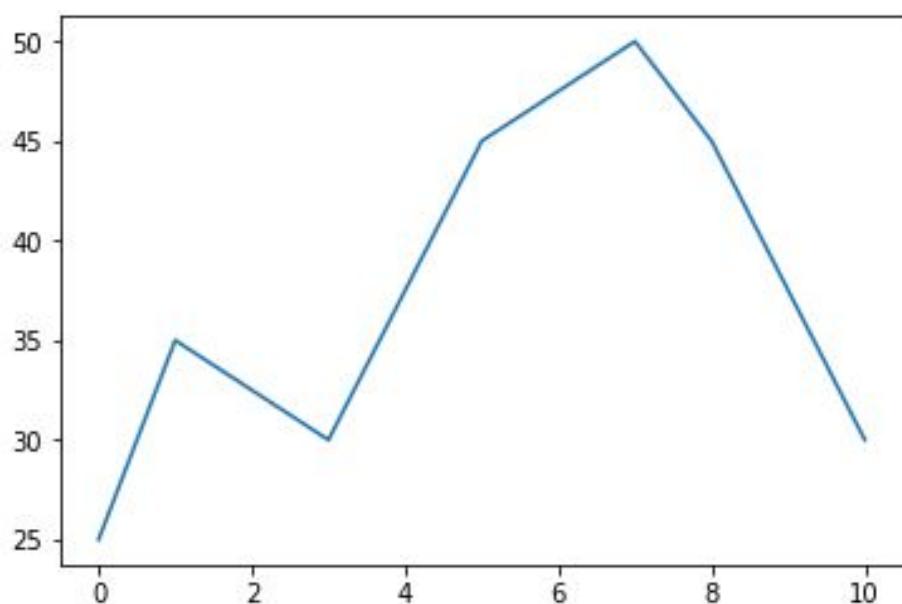


Figure 3.48: Aplikasi Menggunakan Matplotlib

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

Figure 3.49: 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 3.50: Hasil 2 Random Forest

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

Figure 3.51: 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 3.52: 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    ...
2      0    0    0    0    0    0    0    ...
3      0    0    0    0    1    0    0    ...
4      0    0    0    0    1    0    0    ...
5      0    0    0    0    1    0    0    ...
[5 rows x 312 columns]
```

Figure 3.53: Hasil 5 Random Forest

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

Figure 3.54: 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 3.55: Hasil 7 Random Forest

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

```
Out[9]:
```

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

Figure 3.56: Hasil 8 Random Forest

```
In [10]: imglabels.shape
```

```
Out[10]: (3677856, 1)
```

Figure 3.57: 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 3.58: 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 3.59: 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    0    1
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    0    0
```

[5 rows x 312 columns]

Figure 3.60: Hasil 11 Random Forest

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

Figure 3.61: 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 3.62: Hasil 13 Random Forest

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

Figure 3.63: 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 3.64: Hasil 15 Random Forest

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

Figure 3.65: Hasil 16 Random Forest

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

Figure 3.66: Hasil 17 Random Forest

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

Figure 3.67: 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 3.68: 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 3.69: 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 3.70: 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]]
```

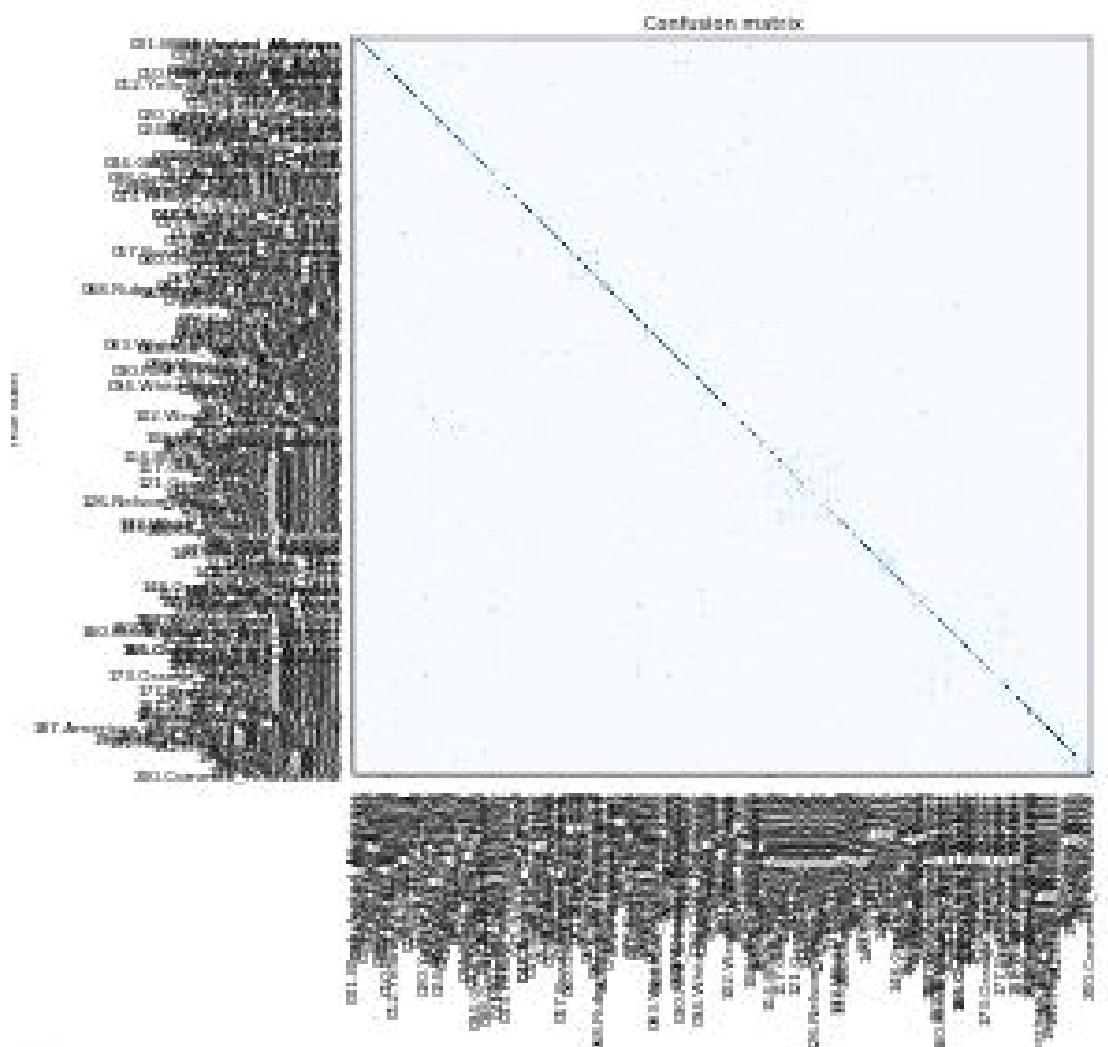


Figure 3.71: Hasil 5 Confusion Matrix

```
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 3.72: 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 3.73: 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 3.74: 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 3.75: Hasil 2 Cross Validation

```

In [28]: scoresvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv=5)
...:     ...: print("Accuracy: %0.2f (+/- %0.2f)" % (scoresvm.mean(), scoresvm.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 3.76: 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: 0.00 (+/- 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 3.77: 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()
```

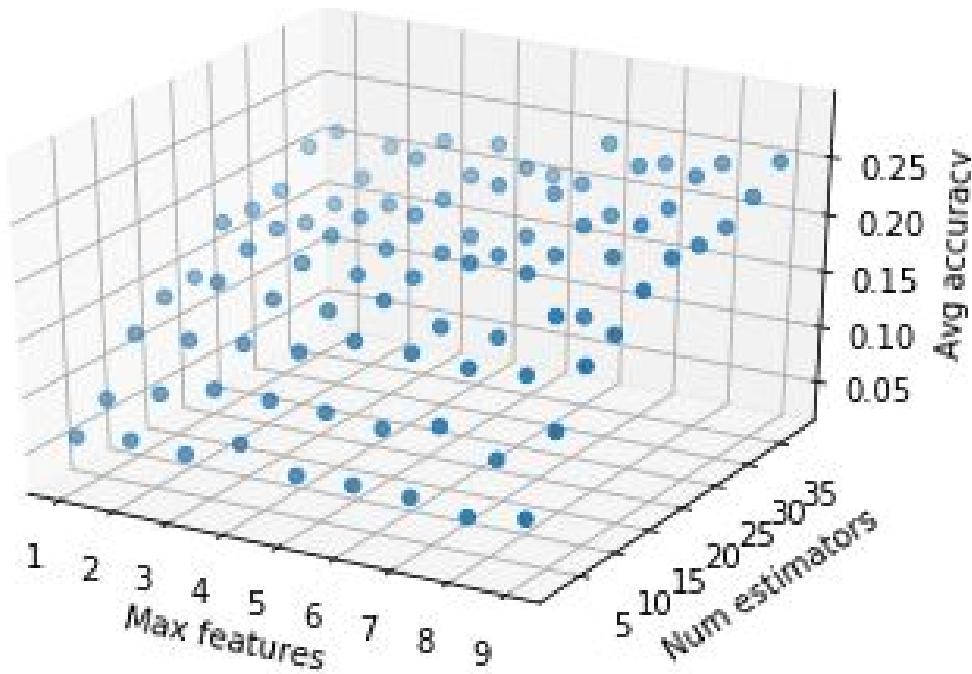


Figure 3.78: Hasil 2 Pengamatan Komponen Informasi

MemoryError

Figure 3.79: Eror

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 Cokro Edi Prawiro / 1164069

4.3.1 Teori

1. Jelaskan apa itu klasifi

kasi teks, sertakan gambar ilustrasi buatan sendiri.

klasifikasi teks adalah cara untuk memilah-milah teks berdasarkan parameter tertentu baik itu jenis teks atau jenis dari dokumen yang terdapat kumpulan teks didalamnya, sedangkan teks itu sendiri merupakan sekumpulan kata yang dapat dibaca. bisa berupa buku, majalah, rambu-rambu dan lain sebagainya. agar lebih jelas dapat dilihat pada gambar 4.1

2. Jelaskan mengapa klasifi

kasi bunga tidak bisa menggunakan machine learning, sertakan ilustrasi sendiri.

Klasifikasi bunga tidak dapat menggunakan mesin learning dikarenakan jenis-jenis bunga banyak yang mirip bahkan banyak bunga yang serupa tetapi tidak

sama. oleh karena itu klasifikasi bunga tidak bisa di gunakan oleh mesin learning dikarenakan jika salah satu inputan ciri-ciri dari siatu bunga di inputkan kemungkinan jawaban dari mesin learning itu tidak tepat contoh dimasukan inputan ciri ciri bunga mawar putih kemudian mesin learning menjawab bahwa itu bunga mawar merah. untuk lebih jelasnya dapat dilihat pada gambar 4.2

3. Jelaskan bagaimana teknik pembelajaran mesin pada teks pada kata-kata yang digunakan di youtube,jelaskan arti per atribut data csv dan sertakan ilustrasi buatan sendiri.

cara pembelajaran teks yang di gunakan youtube yaitu dengan cara merekam data yang sering di inputkan oleh user pada menu pencarian youtube. sehingga pada saat user akan mencari data yang serupa seringkali youtube menyediakan opsi atau rekomendasi-rekomendasi dari pencaharian. contoh saya menuliskan m maka muncul opsi pilihan master chep dan lainya yang berawalan m rekomendasi yang muncul merupakan kata-kata yang sering di cari oleh banyak user atau sering di buka oleh user itu sendiri untuk lebih jelasnya dapat dilihat pada gambar 4.3

4. Jelaskan apa yang dimaksud vektorisasi data.

vektorisasi data merupakan pemecahan data menjadi bagian-bagian yang lebih sederhana contoh pada satu paragraf terdiri dari 200 kata kemudian dilakukan vektorisasi dengan cara membagi-bagi kata dalam paragraf tersebut ke dalam kalimat-kalimat yang terpisah kemudian di pecah lagi menjadi data dalam perkata selanjutnya kata-kata tersebut di terjemahkan.

5. Jelaskan apa itu bag of words dengan kata-kata yang sederhana dan ilustrasi sendiri. bag of words merupakan proses penyederhanaan kata-kata yang asalnya tersirat dalam satu kalimat atau satu paragraf di ubah menjadi perkata kemudian kata-kata tersebut di kumpulkan menjadi satu kelompok tanpa ada arti dari kata-kata yang telah di kumpulkan tersebut lalu di hitung frekuensi kemunculan dari kata tersebut. supaya lebih jelas dapat dilihat pada contoh gambar 4.4 :

6. Jelaskan apa itu TF-IDF, ilustrasikan dengan gambar sendiri. TF-IDF merupakan metode untuk menghitung bobot dari kata yang sering muncul pada suatu kalimat. metode ini menghitung nilai TF atau Term Frequency dan IDF atau Inverse Document Frequency pada setiap kata pada kalimat yang dijadikan

acuan kata pada metode ini sering disebut token adapun rumus dari metode ini dapat dilihat pada gambar 4.5 dan untuk hasil ujicobanya dapat di lihat pada gambar 4.6 dan gambar 4.7.

4.3.2 Praktek Program

1. import data pandas dan 500 baris data dummy kemudian di jelaskan tiap barisnya.

```
# In[1]: mengimport librari padas yang di gunakan
# untuk membaca file tex atau csv
import pandas as pd
#membaca file csv menggunakan fungsi read csv dari padas
data_forest = pd.read_csv("E:/KULIAH/..../AI/Buku/Chapter03/forest.csv")
# In[2]: untuk melihat jumlah dari baris data yang telah di import
len(data_forest)
# In[3]: untuk melihat lima baris pertama data yang telah di import
data_forest.head()
# In[4]: untuk mengetahui banyak baris dan kolom dari data yang
# telah di import.
data_forest.shape
```

2. memecah data frame menjadi dua yang pertama 450 dan kedua sisanya

```
# In[5]: membuat data training dan data testing

# jumlah baris data training sebanyak 450 baris
data_training = data_forest[:450]
# jumlah baris data testing dari hasil pengurangan 523-450
data_testing = data_forest[450:]
```

3. praktik vektorisasi

berikut merupakan codingan untuk melakukan vektorisasi data berupa teks dalam format csv

```
# In[1]:
#melakukan import pandas untuk membaca file csv
import pandas as pd
data_komen=pd.read_csv("E:/.../.../Chapter03/Youtube02-KatyPerry.csv")
```

```

# In[2]:
#mengelompokkan komentar spam dan bukan spam
spam=data_komen.query('CLASS == 1')
nospam=data_komen.query('CLASS == 0')

# In[3]: memanggil lib vektorisasi
#melakukan fungsi bag of word dengan cara menghitung semua kata
#yang terdapat dalam file
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()

# In[3]: memilih kolom CONTENT untuk dilakukan vektorisasi
#melakukan bag of word pada dataframe pada kolom CONTENT
data_vektorisasi = vectorizer.fit_transform(data_komen['CONTENT'])

# In[4]: melihat isi vektorisasi
data_vektorisasi

# In[5]: melihat isi data pada baris ke 349
print(data_komen['CONTENT'][349])

# In[6]: melihat daftar kata yang di vektorisasi
#feature_names merupakan digunakan untuk mengambil nama
#kolomnya ada apa saja
dk=vectorizer.get_feature_names()

# In[7]: akan melakukan randomisasi pada database nya supaya
#sempurna saat melakukan klasifikasi
acak_acak = data_komen.sample(frac=1)

# In[8]: membuat data training dan testing
dk_train=acak_acak[:300]
dk_test=acak_acak[300:]

# In[9]: melakukan training pada data training dan di vektorisasi

```

```

dk_train_att=vectorizer.fit_transform(dk_train['CONTENT'])

dk_train_att

# In[10]: melakukan testing pada data testing dan di vektorisasi
dk_test_att=vectorizer.transform(dk_test['CONTENT'])

dk_test_att

# In[11]: Dimana akan mengambil label spam dan bukan spam
dk_train_label=dk_train['CLASS']
dk_test_label=dk_test['CLASS']

```

untuk hasilnya dapat dilihat pada gambar 4.8 pada gambar tersebut di bagian In [100] dilakukan import library pandas yang di inisialisasi menjadi pd setelah itu ada dibuat class data_komen dengan method read_csv untuk membaca file berekstensikan csv yang di masukan alamatnya pada kurung kemudian pada bagian In [101] dilakukan klasifikasi atau pemilihan komentar yang berisi spam atau bukan spam dengan parameter class samadengan 1 merupakan spam dan class samadengan 0 bukan spam setelah itu pada bagian In [102] memasukan librari CountVektorizer yang digunakan untuk vektorisasi data kemudian dilanjutkan pada bagian In[103] dibuat variabel yang berisi vektorisasi dari data pada data_komen di field content setelah itu variabel tersebut di running pada bagian In[104] yang hasilnya menunjukan 350 baris di kali 1738 kolom selanjutnya pada bagian In[105] dicoba untuk memunculkan isi recod pada baris ke 349 maka akan muncul isian dari baris tersebut. selanjutnya dibuat variabel dk atau daftar pada bagian In[106] yang berisi data hasil vektorisasi setelah itu pada bagian In [107] yang terdiri dari variabel acak_acak yang berisi data komen yang di dalamnya di buat random yang nantinya akan dibut data training dan data testing pada bagian In [108] dengan ketentuan data training 300 dan data testing sebanyak 50 setelah itu data training di lakukan vektorisasi pada bagian In [109] dan data testing juga dilakukan vektorisasi pada bagian In [110] setelah itu kedua data training dan testing tersebut dibuat label pada bagian In [111] dengan parameter field CLASS pada tabel.

4. klasifikasi SVM berikut ini merupakan codingan klasifikasi SVM

```

# In[18] :
from sklearn import svm

```

```
clfsvm = svm.SVC()
clfsvm.fit(dk_train_att, dk_train_label)
clfsvm.score(dk_test_att, dk_test_label)
```

untuk hasil codingan klasifikasi svm tersebut dapat dilihat pada gambar 4.9 pada bagian In [119] melakukan verifikasi import librari svm dari sklearn kemudian membuat variabel clfsvm berisikan method svc setelah itu variabel tersebut di berikan method fit dengan isian data train vektorisasi dan data training label yang berguna untuk melatih data tersebut setelah itu di coba untuk memunculkan score atau akurasi dari data tersebut menggunakan data testing vektorisasi dan data testing label.

5. klasifikasi decision tree berikut ini merupakan codingan klasifikasi decision tree

```
# In[17]:
from sklearn import tree
clftree = tree.DecisionTreeClassifier()
clftree.fit(dk_train_att, dk_train_label)
clftree.score(dk_test_att, dk_test_label)
```

untuk hasil codingan klasifikasi decision tree tersebut dapat dilihat pada gambar 4.10 pada bagian In [118] melakukan verifikasi import librari tree dari sklearn kemudian membuat variabel clftree berisikan method DecisionTreeClasifier setelah itu variabel tersebut di berikan method fit dengan isian data train vektorisasi dan data training label yang berguna untuk melatih data tersebut agar dapat digunakan pada codingan selanjutnya setelah itu di coba untuk memunculkan score atau akurasi dari data tersebut menggunakan data testing vektorisasi dan data testing label.

6. plot confusion matrix berikut merupakan codingan untuk confusion matrix

```
# In[15]: Membuat confusion Matrix dan menampilkannya
from sklearn.metrics import confusion_matrix
pred_labels = clf.predict(dk_test_att)
cm = confusion_matrix(dk_test_label, pred_labels)

cm
```

untuk hasil dari codingan tersebut dapat dilihat pada gambar 4.11 pada bagian In [116] dilakukan import library comfusion matrix selanjutnya dilakukan prediksi pada pada data tes nya kemudian data tersebut di masukan kedalam variabel cm dengan method confusion matrix yang di dalamnya terdapat data dari variabel perd label dan dk test label setelah itu variabel cm tersebut di running maka akan muncul hasil Out[116] memunculkan nilai matrixnya.

7. cross valodation

berikut merupakan code untuk cross validation pada codingan pertama yaitu melakukan split 5 kali yaoti mengitung tingkat akurasi menggunakan data training.

```
# In[16]: Dimana akan melakukan cross validation dengan 5 split
from sklearn.model_selection import cross_val_score
scores = cross_val_score(clf, dk_train_att, dk_train_label, cv=5)
scorerata2=scores.mean()
scorersd=scores.std()

# In[21]:
from sklearn.model_selection import cross_val_score
scores = cross_val_score(clf, dk_train_att, dk_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))
# In[22]:
scorestree = cross_val_score(clftree, dk_train_att, dk_train_label, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(),
scorestree.std() * 2))
# In[23]:
scoressvm = cross_val_score(clfsvm, dk_train_att, dk_train_label, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(),
scoressvm.std() * 2))
```

untuk hasil codingan ke dua dapat dilihat pada gambar 4.12 pada gambar tersebut memunculkan nilai akurasi dari tiga metode yaitu random forest, decision tree, dan klasifikasi svm (suport vector machine) diamana akan di bandingkan

tingkat akurasi dari semua hasil akurasi mana yang terbaik dan lebih akurat pada hasilnya data yang paling akurat yaitu random forest.

8. Pengamatan program

```
# In[24]:  
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, dk_train_att, dk_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))  
  
# In[25]:  
import matplotlib.pyplot as plt  
from mpl_toolkits.mplot3d import Axes3D  
from matplotlib import cm  
fig = plt.figure()  
fig.clf()  
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(0.6, 1)
```

```
ax.set_xlabel('Max features')
ax.set_ylabel('Num estimators')
ax.set_zlabel('Avg accuracy')
plt.show()
```

hasil dari codingan diatas dapat dilihat pada gambar 4.13 pada gambar tersebut terdapat grafik data yang terdapat dari grafik tersebut di dapat dari codingan In[24] dengan cara pengulangan data masing masing 10 kali setelah itu di eksekusi menjadi grafik berbentuk 3D yang dapat di lihat pada gambar 4.14 pada gambar tersebut menunjukan rasio dari yang terrendah yaitu data SVM kemudian data decision tree dan hasil random forest.

4.3.3 Penanganan Error

1. screnn shoot error dapat dilihat pada gambar 4.15
2. codingan yang errornya terdapat pada

```
clfsvm = svm.SVC()
```

3. solusinya

```
clfsvm = svm.SVR(gamma = 'auto')
```

maka hasilnya dapat dilihat pada gambar 4.16

4.4 Fathi Rabbani / 1164074

4.4.1 Teori

1. Klasifikasi Teks

klasifikasi teks adalah sebuah cara dalam memilah data teks berdasarkan beberapa parameter tertentu dengan data yang bersifat dokumen ataupun teks yang memiliki kumpulan - kumpulan teks didalamnya, serta teks itu sendiri merupakan sebuah data yang bisa dibaca oleh manusia atau bisa disebut juga dengan tipe data karakter atau string yang mudah untuk diolah seperti pada buku, majalah, koran, jurnal dan lainya. untuk ilustrasi dari klasifikasi teks bisa dilihat pada gambar 4.17

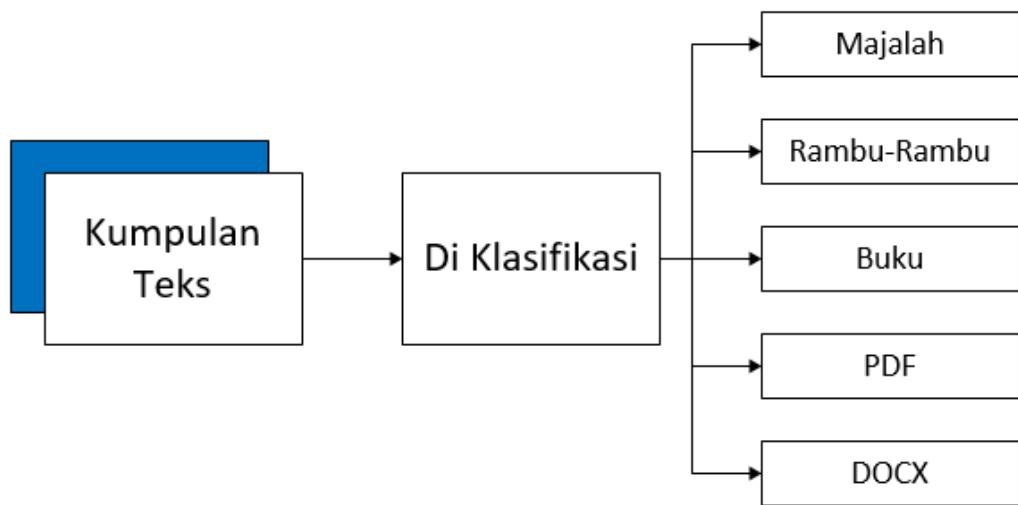


Figure 4.1: Ilustrasi clasifikasi Teks

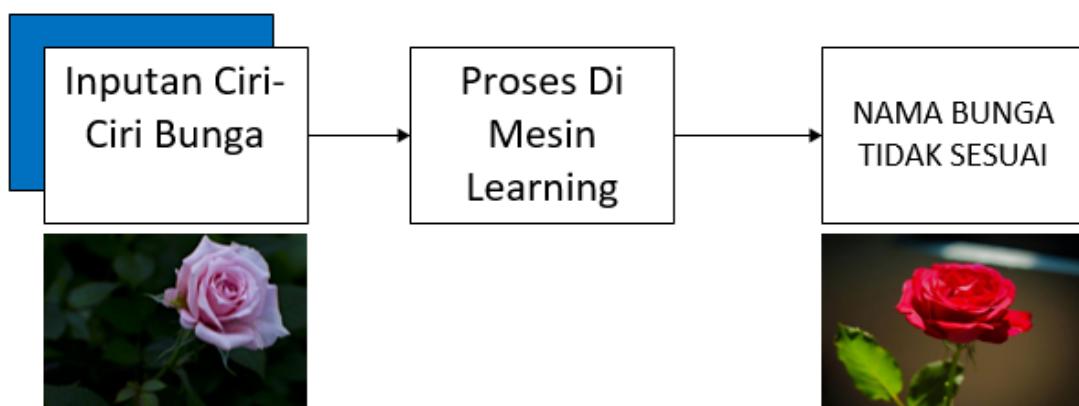


Figure 4.2: Ilustrasi Bunga Tidak bisa dibaca di Mesin Learning

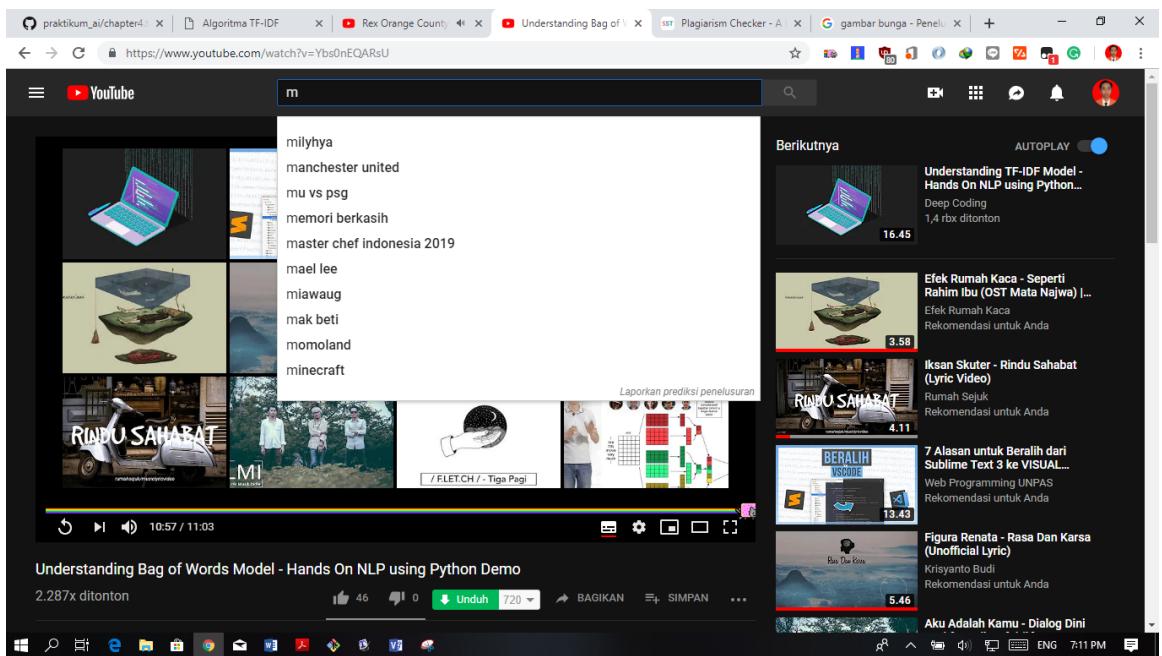


Figure 4.3: contoh hasil pembelajaran text di youtube

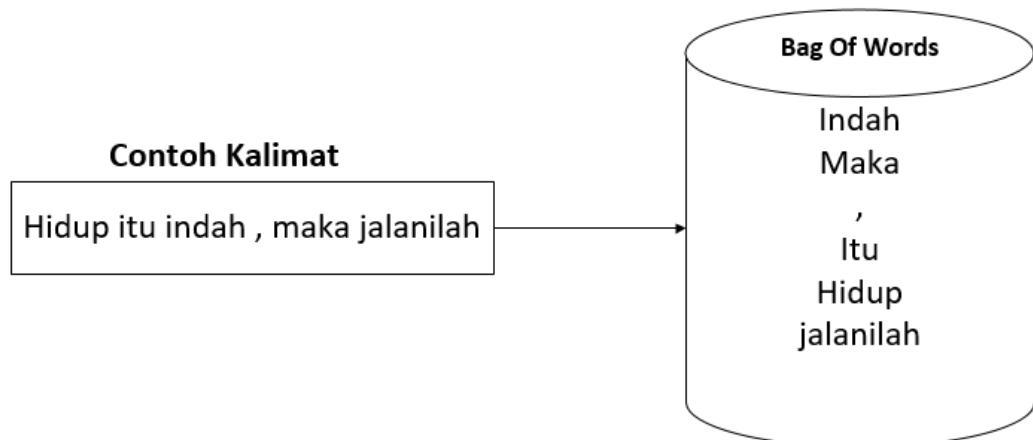


Figure 4.4: Ilustrasi bag of words

$$W_{dt} = tf_{dt} * IDF_t$$

Dimana :

- d : dokumen ke-d
- t : kata ke-t dari kata kunci
- W : bobot dokumen ke-d terhadap kata ke-t
- tf : banyaknya kata yang dicari pada sebuah dokumen
- IDF : Inversed Document Frequency

Nilai IDF didapatkan dari $IDF = \log_2(D/df)$ dimana :

- D : total dokumen
- df : banyak dokumen yang mengandung kata yang dicari

Figure 4.5: Rumus TF-IDF

2. Mengapa Klasifikasi Bunga tidak dapat menggunakan Machine Learning

klasifikasi menggunakan tipe data yang dimana attributnya memiliki nilai data berupa vektor dengan perbandingan masing - masing data yang dimiliki memiliki sedikit perbedaan, sehingga program atau sistem tidak dapat membedakan dengan tepat antara gambar 1 dan gambar 2 dikarenakan memiliki perbedaan yang hampir tidak dapat dilihat pada beberapa contoh gambar. untuk ilustrasi dapat dilihat pada gambar 4.18

3. Teknik Pembelajaran Youtube pada Teks

Youtube merupakan platform untuk berbagi pengalaman video para penggunanya, dan youtube menggunakan machine learning dengan membaca data yang penggunanya inputkan disebut juga temporary data, yang menyimpan data pencarian para penggunanya dengan membuat kolom klasifikasinya seperti pada gambar 4.19, dimana data pencarian saya memiliki 6 data dengan jumlah atribut yang masuk adalah 15.

4. Vektorisasi Data

| 1. Hapus Karakter Khusus | | | | |
|---|--------------------------|-------------|----|------------------|
| Teks 1 | Teks 2 | | | |
| saya ke jebak hujan kemarin | kemarin hujan saya galau | | | |
| 2. Tentukan bobot untuk setiap term dari dokumen-dokumen tersebut | | | | |
| Term | TF | IDF | | |
| | Teks 1 (Q) | Teks 2 (D1) | df | log(n/df) |
| saya | 1 | 1 | 2 | 0 |
| ke | 1 | 0 | 1 | 0.30102999566398 |
| jebak | 1 | 0 | 1 | 0.30102999566398 |
| hujan | 1 | 1 | 2 | 0 |
| kemarin | 1 | 1 | 2 | 0 |
| galau | 0 | 1 | 1 | 0.30102999566398 |

| 3. Hitung Wdt=tf.idf | |
|----------------------|------------------|
| Q | D1 |
| 0 | 0 |
| 0.30102999566398 | 0 |
| 0.30102999566398 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0.30102999566398 |

Figure 4.6: Hasil Perhitungan TF-IDF

4. Hitung hasil perkalian skalar antara Q dan dokumen lain

| |
|-------------------|
| WD*Wd1 |
| D1 |
| 0 |
| 0.090619058289457 |
| 0.090619058289457 |
| 0 |
| 0 |
| 0 |

5. Hitung panjang setiap dokumen, termasuk Q

| Panjang Vektor | |
|-------------------|-------------------|
| Q | D1 |
| 0 | 0 |
| 0.090619058289457 | 0 |
| 0.090619058289457 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0.090619058289457 |

6. Terapkan rumus cosine similarity

| |
|------------------------|
| WD*Wd1 (Total) |
| D1 |
| 0 |
| Panjang Vektor (Total) |
| Q |
| 0.42572070254912 |
| D1 |
| 0.30102999566398 |

$$\text{Cos}(Q, D1) = 0$$

Figure 4.7: Hasil Perhitungan TF-IDF

```

In [100]: import pandas as pd
...: data_komen=pd.read_csv("E:/KULIAH/semester_6/AI/Buku/Chapter03/Youtube02-KatyPerry.csv")

In [101]: spam=data_komen.query('CLASS == 1')
...: nospam=data_komen.query('CLASS == 0')

In [102]: from sklearn.feature_extraction.text import CountVectorizer
...: vectorizer = CountVectorizer()

In [103]: data_vektorisasi = vectorizer.fit_transform(data_komen['CONTENT'])

In [104]: data_vektorisasi
Out[104]:
<350x1738 sparse matrix of type '<class 'numpy.int64'>'
  with 5275 stored elements in Compressed Sparse Row format>

In [105]: print(data_komen['CONTENT'][349])
who is going to reach the billion first : katy or taylor ?

In [106]: dk=vectorizer.get_feature_names()

In [107]: acak_acak = data_komen.sample(frac=1)

In [108]: dk_train=acak_acak[:300]
...: dk_test=acak_acak[300:]

In [109]: dk_train_att=vectorizer.fit_transform(dk_train['CONTENT'])
...: dk_train_att
Out[109]:
<300x1568 sparse matrix of type '<class 'numpy.int64'>'
  with 4512 stored elements in Compressed Sparse Row format>

In [110]: dk_test_att=vectorizer.transform(dk_test['CONTENT'])
...: dk_test_att
Out[110]:
<50x1568 sparse matrix of type '<class 'numpy.int64'>'
  with 588 stored elements in Compressed Sparse Row format>

In [111]: dk_train_label=dk_train['CLASS']
...: dk_test_label=dk_test['CLASS']

```

Figure 4.8: Hasil running data vektorisasi

```

In [119]: from sklearn import svm
...: clfsvm = svm.SVC()
...: clfsvm.fit(dk_train_att, dk_train_label)
...: clfsvm.score(dk_test_att, dk_test_label)
C:\ProgramData\Anaconda3\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[119]: 0.8

```

Figure 4.9: Hasil running klasifikasi svm menggunakan data vektorisasi

```

In [118]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(dk_train_att, dk_train_label)
...: clftree.score(dk_test_att, dk_test_label)
Out[118]: 0.94

```

Figure 4.10: Hasil running klasifikasi decision treei

```
In [116]: from sklearn.metrics import confusion_matrix
...: pred_labels = clf.predict(dk_test_att)
...: cm = confusion_matrix(dk_test_label, pred_labels)
...:
...: cm
Out[116]:
array([[26,  0],
       [ 2, 22]], dtype=int64)
```

Figure 4.11: Hasil running cnfusion matrix

```
In [122]: from sklearn.model_selection import cross_val_score
...: scores = cross_val_score(clf, dk_train_att, dk_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: 0.94 (+/- 0.06)

In [123]: scorestree = cross_val_score(clftree, dk_train_att, dk_train_label, cv=5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(), scorestree.std() * 2))
Accuracy: 0.91 (+/- 0.06)

In [124]: scoressvm = cross_val_score(clfsvm, dk_train_att, dk_train_label, cv=5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(), scoressvm.std() * 2))
Accuracy: 0.54 (+/- 0.10)
C:\ProgramData\Anaconda3\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)

In [125]: |
```

Figure 4.12: Hasil Cross Validationi

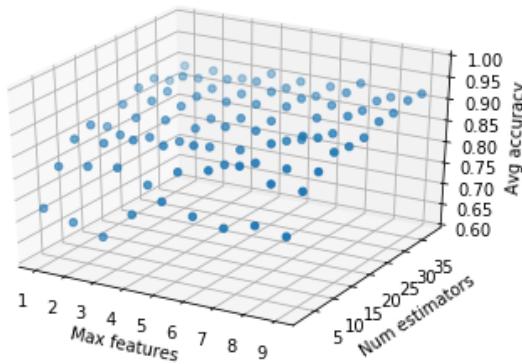
```

In [125]: 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, dk_train_att, dk_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))
Max features: 1, num estimators: 2, accuracy: 0.73 (+/- 0.18)
Max features: 1, num estimators: 6, accuracy: 0.80 (+/- 0.09)
Max features: 1, num estimators: 10, accuracy: 0.85 (+/- 0.11)
Max features: 1, num estimators: 14, accuracy: 0.86 (+/- 0.11)
Max features: 1, num estimators: 18, accuracy: 0.84 (+/- 0.12)
Max features: 1, num estimators: 22, accuracy: 0.87 (+/- 0.08)
Max features: 1, num estimators: 26, accuracy: 0.89 (+/- 0.04)
Max features: 1, num estimators: 30, accuracy: 0.89 (+/- 0.05)
Max features: 1, num estimators: 34, accuracy: 0.87 (+/- 0.08)
Max features: 1, num estimators: 38, accuracy: 0.89 (+/- 0.11)
Max features: 2, num estimators: 2, accuracy: 0.71 (+/- 0.13)
Max features: 2, num estimators: 6, accuracy: 0.82 (+/- 0.10)
Max features: 2, num estimators: 10, accuracy: 0.85 (+/- 0.08)
Max features: 2, num estimators: 14, accuracy: 0.85 (+/- 0.08)
Max features: 2, num estimators: 18, accuracy: 0.87 (+/- 0.10)
Max features: 2, num estimators: 22, accuracy: 0.89 (+/- 0.07)
Max features: 2, num estimators: 26, accuracy: 0.89 (+/- 0.07)
Max features: 2, num estimators: 30, accuracy: 0.91 (+/- 0.03)
Max features: 2, num estimators: 34, accuracy: 0.89 (+/- 0.10)
Max features: 2, num estimators: 38, accuracy: 0.89 (+/- 0.06)
Max features: 3, num estimators: 2, accuracy: 0.69 (+/- 0.11)
Max features: 3, num estimators: 6, accuracy: 0.83 (+/- 0.09)
Max features: 3, num estimators: 10, accuracy: 0.87 (+/- 0.07)
Max features: 3, num estimators: 14, accuracy: 0.86 (+/- 0.16)
Max features: 3, num estimators: 18, accuracy: 0.88 (+/- 0.10)
Max features: 3, num estimators: 22, accuracy: 0.89 (+/- 0.08)
Max features: 3, num estimators: 26, accuracy: 0.89 (+/- 0.07)
Max features: 3, num estimators: 30, accuracy: 0.89 (+/- 0.07)

```

Figure 4.13: pengulangan Isi Grafik

```
In [126]: import matplotlib.pyplot as plt
....: from mpl_toolkits.mplot3d import Axes3D
....: from matplotlib import cm
....: fig = plt.figure()
....: fig.clf()
....: 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(0.6, 1)
....: ax.set_xlabel('Max features')
....: ax.set_ylabel('Num estimators')
....: ax.set_zlabel('Avg accuracy')
....: plt.show()
```



In [127]: |

Figure 4.14: Grafik 3D

```
In [119]: from sklearn import svm
....: clfsvm = svm.SVC()
....: clfsvm.fit(dk_train_att, dk_train_label)
....: clfsvm.score(dk_test_att, dk_test_label)
C:\ProgramData\Anaconda3\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[119]: 0.8
```

Figure 4.15: Gambar Warning

```
In [127]: from sklearn import svm
....: clfsvm = svm.SVC(gamma = 'auto')
....: clfsvm.fit(dk_train_att, dk_train_label)
....: clfsvm.score(dk_test_att, dk_test_label)
Out[127]: 0.8
```

Figure 4.16: Hasil penanganan Error

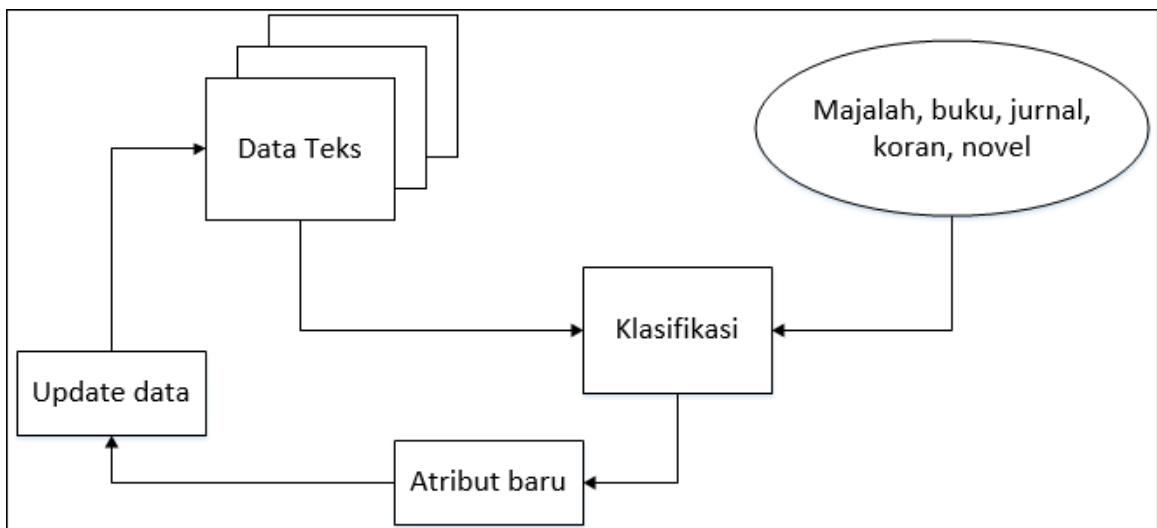


Figure 4.17: Klasifikasi Tekst

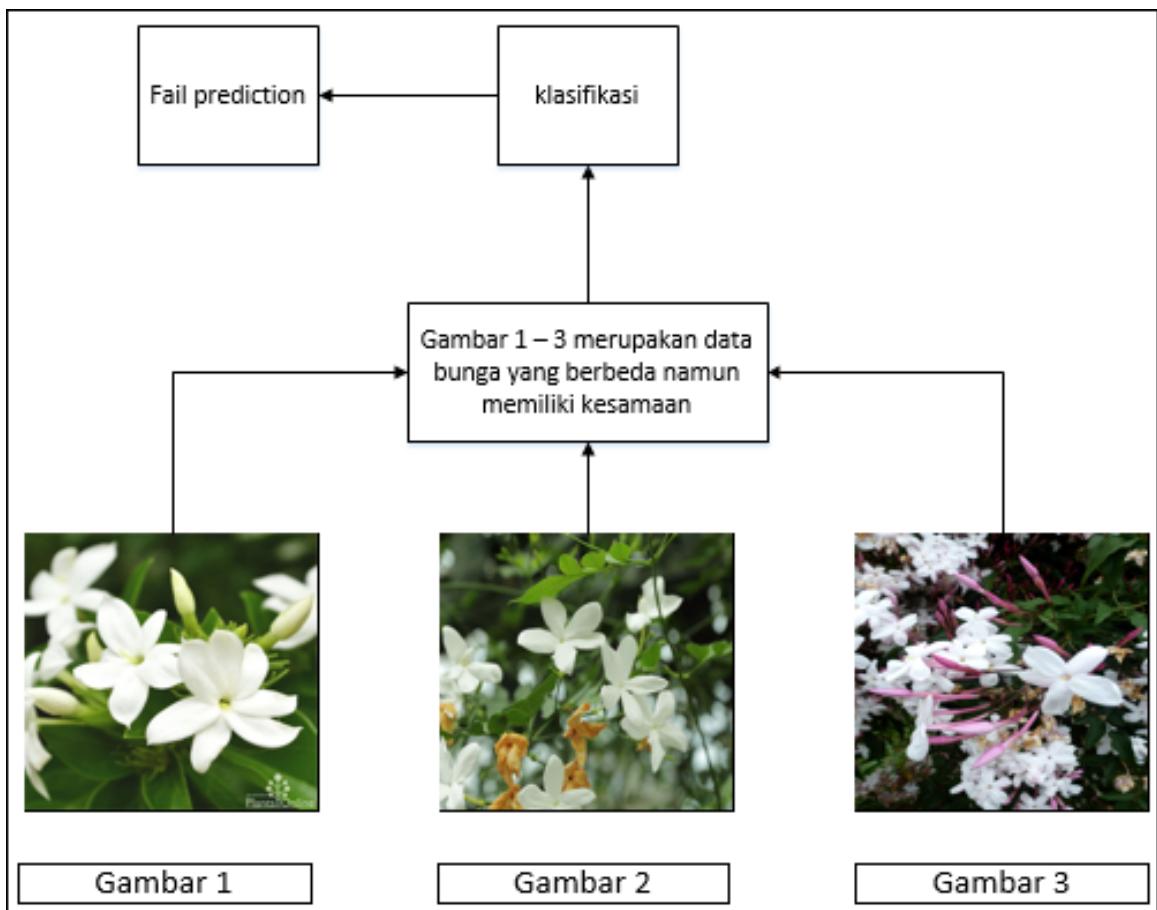


Figure 4.18: Klasifikasi Bunga

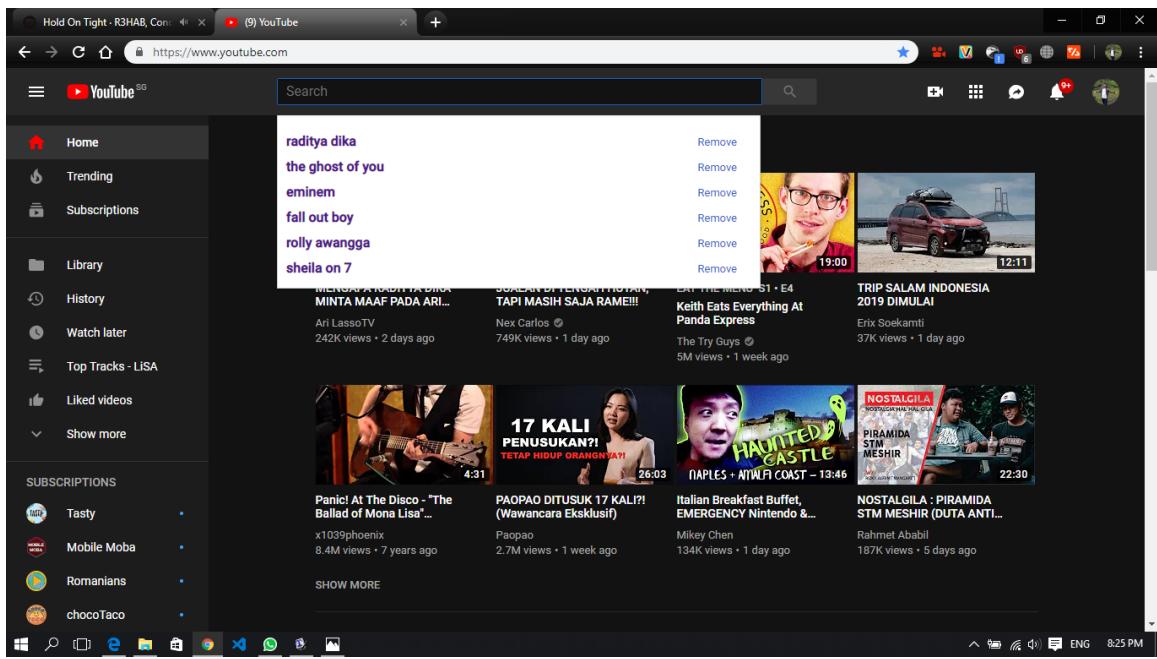


Figure 4.19: Data Youtube

vektor adalah tipe data yang digunakan untuk mempelajari data gambar agar mudah untuk diproses serta lebih sederhana dalam hasil yang dikeluarkan, dan data yang telah divektorisasi terpecah - pecah menjadi beberapa atribut agar mudah untuk diproses.

5. Bag of Words

kumpulan data teks yang diproses dan dibuatkan atribut barunya jika data yang dimiliki sudah tersedia, pada gambar 4.20, menjelaskan jika data yang sama tidak akan diproses menjadi atribut baru sehingga data tersebut sudah disimpan dan akan memproses data baru yang belum dimiliki.

6. TF-IDF

merupakan sebuah metode dalam sebuah perhitungan untuk menghitung bobot dari setiap kata - kata yang sering muncul dalam sebuah kalimat. metode ini sendiri menghitung data nilai dari TF(term of frequency) dan data IDF(inverse document frequency) yang dijadikan sebagai data yang dapat diproses, seperti berikut ini :

- proses pertama adalah dengan membuat data yang akan diolah seperti pada data

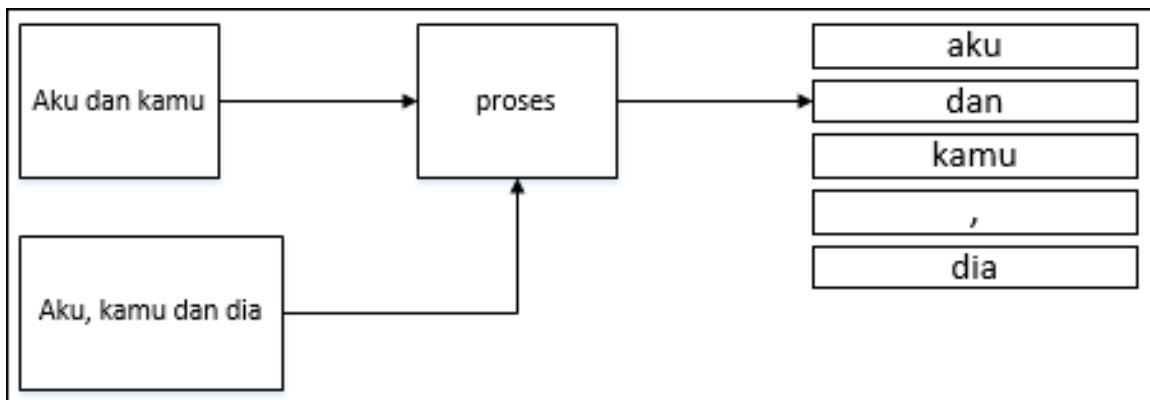


Figure 4.20: Bags of Word

gold
silver
truck

Untuk koleksi dokumennya terdapat:

dokumen 1 (d1) = Shipment of gold damaged in a fire
 dokumen 2 (d2) = Delivery of silver arrived in a silver truck
 dokumen 3 (d3) = Shipment of gold arrived in a truck

Jadi total jumlah dokumen adalah koleksi dokumen (D) = 3

- proses kedua adalah menggabungkan ke 3 data tersebut dengan menghilangkan nilai yang di ulang (term of frequency)-nya dan hasilnya adalah seperti pada

ship gold damage fire deliver silver arrive truck

- kedua proses tersebut dinamakan preprocessing text, dan berikut ini adalah formula yang digunakan untuk menghitung nilai pada bobot data d2 pada

$$W_{ij} = TF_{ij} \times \log(D/DF_j) + 1$$

$$W_{ij} = 2 \times (\log(3/1) + 1)$$

$$W_{ij} = 2 \times (0.477 + 1)$$

$$W_{ij} = 2.954$$

- dengan nilai yang dimiliki seperti itu maka untuk hasilnya adalah seperti pada gambar 4.21

| Q | tf | | | df | D/df | IDF | IDF+1 | W=tf*(IDF+1) | | |
|-----------------------------------|----|----|----|----|------|-------|-------|--------------|---------|---------|
| | d1 | d2 | d3 | | | | | d1 | d2 | d3 |
| gold | 1 | 0 | 1 | 2 | 1.5 | 0.176 | 1.176 | 1.176 | 0 | 1.176 |
| silver | 0 | 2 | 0 | 1 | 3 | 0.477 | 1.477 | 0 | 2.954 | 0 |
| truck | 1 | 1 | 1 | 2 | 1.5 | 0.176 | 1.176 | 0 | 1.176 | 1.176 |
| | | | | | | | | sum(d1) | sum(d2) | sum(d3) |
| Nilai Bobot tiap Dokumen = | | | | | | | | 1.176 | 4.13 | 2.352 |

Figure 4.21: TF-IDF

4.4.2 Praktek

1. Library Pandas dengan Dataset 500

```
# In[1]
# import library pandas dengan menggunakan alias coba1
import pandas as coba1

#membuat variable dengan nama datasets dengan perintah code read\csv dan mem
datasets = coba1.read\csv("C:/../Dokuments/Datasets/forestfire.csv")

#menggunakan perintah len untuk menampilkan hasil data variable
len(datasets)
```

2. Bagi data 450 baris dan 50 baris

```
# In[2]
#membuat baris data menjadi 450
set_train = datasets[:450]

#membuat baris data menjadi 50
set_test = datasets[450:]
```

3. Decision Tree dengan Vektor dan Klasifikasi

berikut ini adalah kode ayng digunakan dan beberapa penjelasannya, untuk hasilnya dapat dilihat pada gambar 6.10

```
# In[3]
#import library pandas dengan alias coba2
import pandas as coba2
```

```

#mengambil datasets csv dengan nama Youtube04-Eminem
datanya = coba2.read\csv("C:/../Dokuments/Datasets/Youtube04-Eminem.csv")

# In[4]
#memanggil library Vektorisasi
from sklearn.feature_extraction.text import CountVectorizer

#menggunakan library vektorisasi pada variable vektor
vektor = CountVectorizer()

# In[5]
#membuat variable dengan perintah untuk memproses kolom CONTENT pada dataset
varvektor = vektor.fit_transform(datanya['CONTENT'])

#membaca variable data
varvektor

# In[6]
#menggunakan randomizer untuk membuat datanya menjadi teratur
absurd = datanya.sample(frac=1)

# In[7]
#membuat data train dan test
set_train = absurd[:300]
set_test = absurd[300:]

# In[8]
#melakukan test pada data train dengan vektorisasi
set_train_att = vektor.fit_transform(set_train['CONTENT'])
set_train_att

# In[9]
#melakukan test pada data test dengan vektorisasi
set_test_att = vektor.fit_transform(set_test['CONTENT'])
set_test_att

```

```
# In[10]
#mengambil data CLASS dan dijadikan data train dan test label
set_train_label = set_train['CLASS']
set_test_label = set_test['CLASS']
```

4. Klasifikasi SVM

berikut adalah kode yang digunakan dan beberapa penjelasannya, untuk hasilnya bisa dilihat pada gambar 6.11

```
# In[14]
#memanggil library SVM dari sklearn
from sklearn import svm

#membuat data variable dengan library SVM dan perintah SVC
clfsvm = svm.SVC()

#memasukkan data attribute train dan label train kedalam variable clfsvm
clfsvm.fit(set_train_att, set_train_label)

#melihat skor dari data attribute test dan test label
clfsvm.score(set_test_att, set_test_label)
```

5. klasifikasi Decision Tree

berikut ini adalah kode yang digunakan dan beberapa penjelasannya untuk hasilnya dapat dilihat pada gambar 6.12

```
# In[15]
#memanggil library decision tree dari sklearn
from sklearn import tree

#membuat data variable dengan library decision tree dan perintah kodennya adalah
clftree = tree.DecisionTreeClassifier()

#memasukkan data attribute train dan label train kedalam variable clftree
clftree.fit(set_train_att, set_train_label)
```

```
#melihat skor dari data attribute test dan test label  
clftree.score(set_test_att, set_test_label)
```

6. Confusion matrix dan matplotlib

berikut ini adalah kode yang digunakan dan beberapa penjelasannya untuk hasil dari penggunaan confusion matrix dapat dilihat pada gambar 6.13 dan data penggunaan confusion matrix dengan matplotlib dapat dilihat pada gambar 6.14

```
# In[20]  
#mengambil data library confusion matrix dari sklearn  
from sklearn.metrics import confusion_matrix  
  
#membuat variable data dengan isi perintahnya untuk memprediksi nilai dari at  
siap_label = clf.predict(set_test_att)  
  
#membuat variable dengan perintah datanya menggunakan library confusion matrix  
cm = confusion_matrix(set_test_label, siap_label)  
  
#membaca data variable  
cm  
  
  
# In[21]  
#memanggil library matplotlib dengan dialiaskan menjadi plt  
import matplotlib.pyplot as plt  
  
#membuat fungsi untuk membaca data confusion matrix yang dibuat tadi  
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')

#menampilkan hasil data dari penggunaan matplotlib pada confusion matrix
print(cm)

# In[22]
#memanggil library numpy dan dialiaskan menjadi np
import numpy as np

#library np diperintahkan untuk membaca data dengan nilai presisinya 2
np.set_printoptions(precision=2)

#membaca data plot confusion matrixnya dengan nilai data yang dibaca adalah c
plot_confusion_matrix(cm, classes=datanya, normalize=True)

#menampilkan data yang telah diolah pada confusion matrix dan diproses menggu
plt.show()

```

7. klasifikasi Cross Validation

berikut ini adalah kode yang digunakan dan beberapa penjelasannya untuk hasilnya dapat dilihat pada gambar 6.15, 6.16 dan 6.17

```

# In[24]
#memanggil library cross validasi dari sklearn dengan perintah prosesnya mode
from sklearn.model_selection import cross_val_score

#membuat variable SCORES yang menampung data clf, atribut train dan label tra
scores = cross_val_score(clf, set_train_att, set_train_label, cv=5)

# show average score and +/- two standard deviations away (covering 95% of sc
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

```

```

# In[25]
#menampilkan skor untuk proses decision tree menggunakan cross validasi
scorestree = cross_val_score(clftree, set_train_att, set_train_label, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(), scorestree.std() * 2))

# In[26]
#menampilkan skor untuk proses SVM menggunakan cross validasi
scoressvm = cross_val_score(clfsvm, set_train_att, set_train_label, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(), scoressvm.std() * 2))

#membuat data variable yang akan menunjang proses yang akan digunakan sebagai
dengan menampilkan keseluruhan nilai dari skor yang ada, mulai dari cross val
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, set_train_att, set_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, rf_params[i-1,2], rf_params[i-1,3]))

```

8. Pengamatan Program

berikut ini adalah kode yang digunakan dan beberapa penjelasannya untuk hasilnya dapat dilihat pada gambar 6.18

```

# In[28]
#memanggil library matplotlib dengan dialiaskan menjadi plt
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d, Axes3D

```

```
from matplotlib import cm
fig = plt.figure()
fig.clf()
ax = Axes3D(fig)
x = rf_params[:,0]
y = rf_params[:,1]
z = rf_params[:,2]
ax.scatter(x, y, z)
ax.set_zlim(0.9, 1)
ax.set_xlabel('Max features')
ax.set_ylabel('Num estimators')
ax.set_zlabel('Avg accuracy')
plt.show()
```

4.4.3 Error

1. eror pada gambar 4.31
2. error pada code

```
ax = fig.gca(projection='3d')
```

3. solusi

```
from mpl_toolkits.mplot3d import axes3d, Axes3D
```

| Name | Type | Size | Value |
|-----------------|-----------|----------|--|
| absurd | DataFrame | (448, 5) | Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS |
| bukanspamming | DataFrame | (203, 5) | Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS |
| datanya | DataFrame | (448, 5) | Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS |
| set_test | DataFrame | (148, 5) | Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS |
| set_test_label | Series | (148,) | Series object of pandas.core.series module |
| set_train | DataFrame | (300, 5) | Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS |
| set_train_label | Series | (300,) | Series object of pandas.core.series module |
| spamming | DataFrame | (245, 5) | Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS |
| wk | list | 1602 | ['00', '000', '047000', '09', '10', '100', '1000', '100877300245414', ...] |

Figure 4.22: Decision Tree di Vektorisasikan

```
In [78]: from sklearn import svm
....: clfsvm = svm.SVC()
....: clfsvm.fit(set_train_att, set_t
....: clfsvm.score(set_test_att, set_
C:\Users\Fathi-PC\Anaconda3\lib\site-pac
The default value of gamma will change f
account better for unscaled features. Se
avoid this warning.
"avoid this warning.", FutureWarning)
Out[78]: 0.9054054054054054
```

Figure 4.23: klasifikasi SVM

```
In [79]: from sklearn import tree
....: clftree = tree.DecisionTreeClassifier()
....: clftree.fit(set_train_att, set_train_label)
....: clftree.score(set_test_att, set_test_label)
Out[79]: 0.9594594594594594
```

Figure 4.24: klasifikasi Tree

```
In [84]: from sklearn.metrics import confusion_matrix
....: siap_label = clf.predict(set_test_att)
....: cm = confusion_matrix(set_test_label, siap_label)
....: cm
Out[84]:
array([[67,  0],
       [ 8, 73]], dtype=int64)
```

Figure 4.25: Confusion matrix

```
In [87]: import numpy as np
....: np.set_printoptions(precision=2)
....: plot_confusion_matrix(cm, classes=datanya, normalize=True)
....: plt.show()
Normalized confusion matrix
[[1.  0. ]
 [0.1 0.9]]
```

Figure 4.26: matplotlib

```
In [88]: from sklearn.model_selection import cross_val_score
....: scores = cross_val_score(clf, set_train_att, set_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: 0.94 (+/- 0.06)
```

Figure 4.27: Cross validasi

```
In [89]: scorestree = cross_val_score(clftree, set_train_att, set_train_label, cv=5)
....: print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(), scorestree.std() *
2))
Accuracy: 0.94 (+/- 0.06)
```

Figure 4.28: Cross validasi pada tree

```
In [90]: scoressvm = cross_val_score(clfsvm, set_train_att, set_train_label, cv=5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(), scoressvm.std() *
2))
Accuracy: 0.73 (+/- 0.31)
```

Figure 4.29: Cross validasi pada SVM

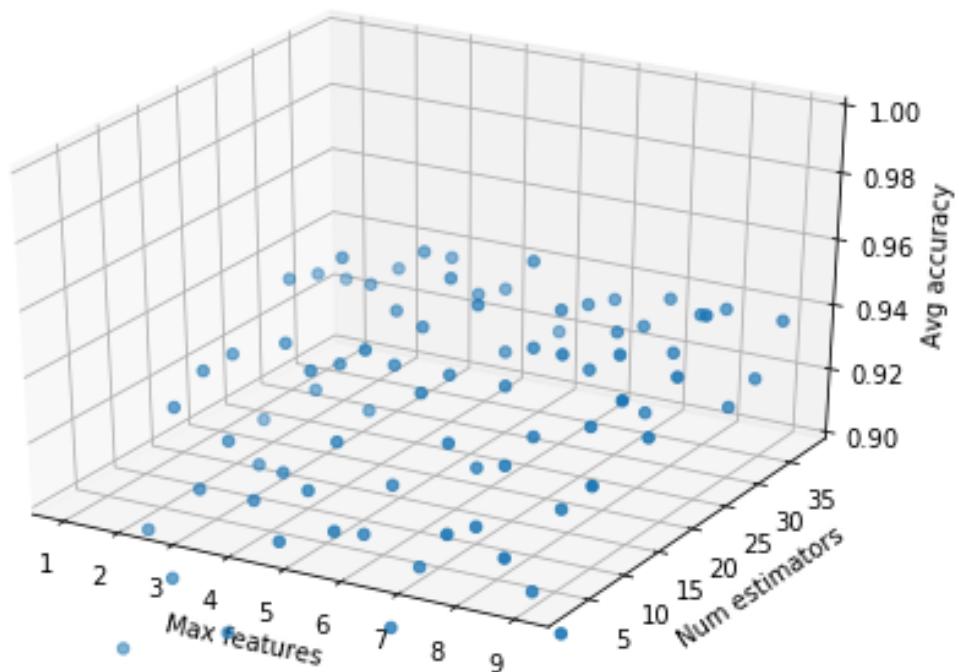


Figure 4.30: pengamatan program

```

In [93]: import matplotlib.pyplot as plt
....: #from mpl_toolkits.mplot3d import Axes3D
....: from matplotlib import cm
....: fig = plt.figure()
....: fig.clf()
....: 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(0.9, 1)
....: ax.set_xlabel('Max features')
....: ax.set_ylabel('Num estimators')
....: ax.set_zlabel('Avg accuracy')
....: plt.show()
Traceback (most recent call last):

File "<ipython-input-93-6d74537c9e62>", line 6, in <module>
    ax = fig.gca(projection='3D')

File "C:\Users\Fathi-PC\Anaconda3\lib\site-packages\matplotlib\figure.py", line
1863, in gca
    return self.add_subplot(1, 1, 1, **kwargs)

File "C:\Users\Fathi-PC\Anaconda3\lib\site-packages\matplotlib\figure.py", line
1349, in add_subplot
    self, *args, **kwargs)

File "C:\Users\Fathi-PC\Anaconda3\lib\site-packages\matplotlib\projections
\_init__.py", line 81, in process_projection_requirements
    projection_class = get_projection_class(projection)

File "C:\Users\Fathi-PC\Anaconda3\lib\site-packages\matplotlib\projections
\_init__.py", line 60, in get_projection_class
    raise ValueError("Unknown projection %r" % projection)

ValueError: Unknown projection '3D'

<Figure size 432x288 with 0 Axes>

```

Figure 4.31: Error

Chapter 5

Conclusion

brief of conclusion

5.1 Cokro Edi Prawiro / 1164069

5.1.1 Teori

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

Kata kata harus dilakukan vektorisasi dikarenakan atau bertujuan utk mengukur nilai kemunculan suatu kata yang serupa dari sebuah kalimat sehingga kata-kata tersebut dapat di prediksi kemunculannya. atau juga di buatnya vektorisasi data digunakan untuk memprediksi bobot dari suatu kata misalkan ayam dan kucing sama-sama hewan maka akan dibuat prediksi apakah kata tersebut akan muncul pada kalimat yang kira-kira memiliki bobot yang sama. untuk dapat jelasnya dapat melihat ilustrasi pada gambar 5.1.

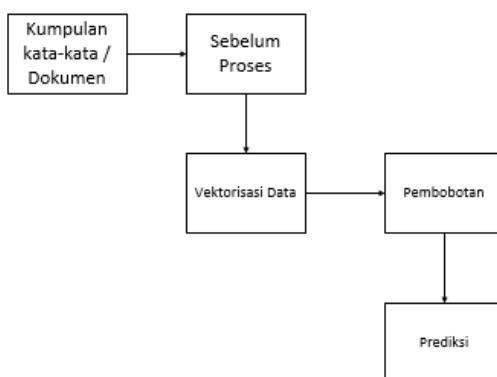


Figure 5.1: Ilustrasi Vektorisasi Kata-Kata

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

Dimensi dataset dari google bisa mencapai 300 karena dimensi dari vektor tersebut digunakan untuk membandingkan bobot dari setiap kata, misalkan terdapat kata dog dan cat pada dataset google tersebut setiap kata tersebut dibuat dimensi vektor 300 untuk kata dog dan 300 dimensi vektor juga untuk kata cat kemudian kata tersebut dibandingkan bobot kesamaan katanya maka akan muncul akurasi sekitar 70 persen kesamaan bobot dikarenakan kata dog dan cat sama-sama digunakan untuk hewan priharaan. Untuk lebih jelasnya dapat dilihat pada gambar 5.2.

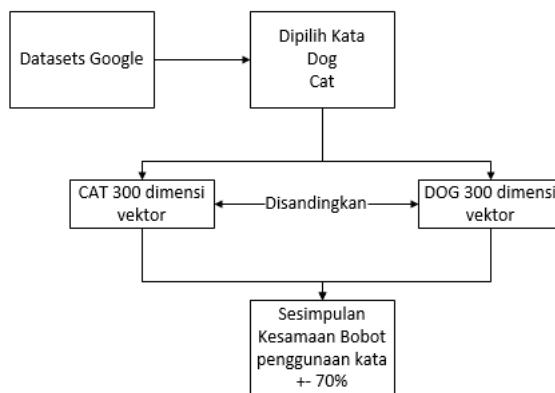


Figure 5.2: Ilustrasi Kenapa dimensi vektor pada datasets google harus 300

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

Vektorisasi untuk kata untuk mengetahui kata tengah dari suatu kalimat atau kata utama atau objek utama pada suatu kalimat contoh (Jangan lupa subscribe channel saya ya sekian terimakasih) kata tengah tersebut merupakan channel 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 dimensiasi dari kata tersebut. Untuk lebih jelasnya dapat dilihat pada gambar 5.3 berikut :

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

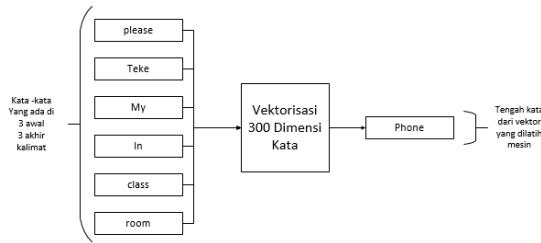


Figure 5.3: Ilustrasi konsep vektorisasi untuk kata

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 contoh dapat dilihat pada gambar 5.4 berikut :

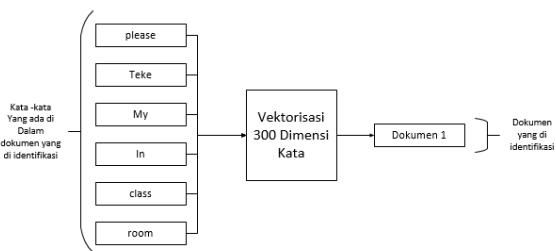


Figure 5.4: Ilustrasi konsep vektorisasi untuk dokumen

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

mean merupakan petunjuk terhadap kata-kata yang diolah jika kata-kata itu akurasinya tinggi berarti kata tersebut sering muncul begitu juga sebaliknya untuk lebih jelasnya dapat dilihat pada gambar 5.5 sedangkan setandar defiation merupakan standar untuk menimbang kesalahan. sehingga kesalahan tersebut dianggap wajar misalkan kita memperkirakan kedalaman dari dataset merupakan 2 atau 3 tapi pada kenyataannya merupakan 5 itu merupakan kesalahan tapi masih bisa dianggap wajar karena masih mendekati perkiraan awal.

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

Skip-Gram adalah kebalikan dari konsep vektorisasi untuk kata dimana kata tengah menjadi acuan terhadap kata-kata pelengkap dalam suatu kalimat untuk lebih jelasnya dapat dilihat pada gambar 5.6 berikut :

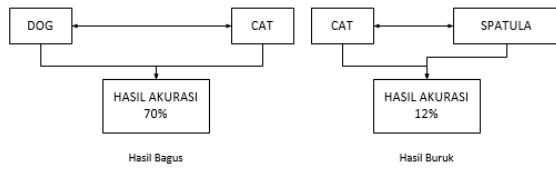


Figure 5.5: Ilustrasi Penggunaan Mean

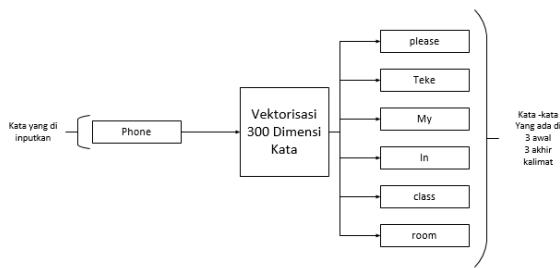


Figure 5.6: Ilustrasi Skip-Gram

5.1.2 Praktikum

1. mencoba dataset google dan penjelasan vektor dari kata love, faith, fall, sick, clear, shine, bag, car, wash, motor, dan cycle.
 - berikut merupakan code import gensim digunakan untuk membuat data model atau rangcangan data yang akan di buat. selanjutnya dibuat variabel gmodel yang berisi data vektor negativ. selanjutnya data tersebut di load agar data tersebut dapat di tampilkan dan di olah. code lengkapnya dapat dilihat pada gambar 5.7.

```
In [27]: import gensim
In [28]: gmodel =
gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-
negative300.bin', binary = True)
```

Figure 5.7: Import Gensim dan membuat model gmodel

- berikut merupakan hasil pengolahan kata clear pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut. untuk jelasnya dapat di lihat pada gambar 5.8.
- berikut merupakan hasil pengolahan kata Shine pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut. untuk jelasnya dapat di lihat pada gambar 5.9.

```
In [11]: gmodel['clear']
Out[11]:
array([-2.44140625e-04, -1.02050781e-01, -1.49414062e-01, -4.24804688e-02,
       -1.67986750e-01, -1.46484375e-01, 1.76757812e-01, 1.46484375e-01,
       -1.24511713e-01, 9.76812500e-02, -2.67578125e-01, -1.08587500e-02,
       1.24511713e-01, 2.46539375e-02, -2.03593750e-01, -1.08587500e-02,
       2.00195312e-01, -1.76074219e-02, -2.03593750e-02, -1.21097575e-01,
       3.22265625e-02, 3.14453125e-01, -1.11816409e-01, 8.00781250e-01,
       -2.75878906e-02, -6.04248847e-03, -7.37394688e-02, -1.72851562e-01,
       9.66796875e-02, -4.91333008e-03, -1.78718938e-01, -1.40380859e-03,
       7.91015625e-02, 1.07910156e-01, -1.10351562e-01, -8.34969938e-02,
       1.98974690e-02, -3.14331055e-03, 1.30615234e-02, 3.34472656e-02,
       2.55859375e-01, -7.12890625e-02, 2.83203125e-01, -2.75390625e-01,
       -8.49609375e-02, -3.73535156e-02, -9.17968750e-02, -1.30859375e-01,
       3.49121094e-02, 7.32421875e-02, 2.17773438e-01, 5.00488281e-02,
       7.47070312e-02, -1.25000000e-01, -5.73730469e-02, -2.74658283e-02,
       -1.37329102e-02, -2.13867188e-01, -1.12792969e-01, -4.98046875e-02,
       -1.92382182e-01, 1.32812500e-01, -5.00488281e-02, -1.66015625e-01,
```

Figure 5.8: Hasil Matrix Clear

```
In [12]: gmodel['shine']
Out[12]:
array([-0.12402344, 0.25976562, -0.15917969, -0.27734375, 0.30273438,
       0.09960938, 0.39257812, -0.22949239, -0.18359375, 0.3671875 ,
       -0.10302734, 0.13671875, 0.25390625, 0.07128906, 0.02539062,
       0.21777344, 0.24023438, 0.5234375 , 0.12394688, -0.19335938,
       -0.05883789, 0.0612795 , 0.01949918, 0.07617893, 0.05102539,
       0.20000000, 0.38828125, 0.08828125, -0.06429579, 0.01848438,
       -0.24765625, -0.2953477 , -0.23932791, 0.04516601, -0.08429579,
       -0.24121094, -0.18945312, -0.15234375 , -0.05693164, 0.01434326,
       0.390625 , -0.2109375 , -0.1484375 , -0.13183594, 0.24511719,
       -0.24023438, -0.36132812, -0.12729869, 0.18595783, 0.09912109,
       -0.24265682, 0.32226562, 0.11376953, 0.18164062, 0.04931641,
       -0.10253906, -0.00283813, -0.29982812, -0.171875 , -0.18945312,
       -0.01367188, -0.20898438, -0.07861328, -0.07859375, 0.05395508,
       -0.14257812, -0.140625 , 0.03927344, -0.14453125, 0.359375 ,
       0.16113281, 0.22225625, 0.265625 , -0.06347465, -0.02887617,
       0.04760742, 0.08837891, -0.04272461, 0.05980203, 0.07128906,
       0.01519775, -0.11621894, 0.07128906, 0.01403809, -0.19644531,
       0.08886719, 0.11523438, 0.09667969, -0.1183984, 0.16015625,
       0.3359375 , -0.1875 , 0.14550781, 0.00463867, 0.07617188,
       -0.09521484, 0.08447266, 0.20117188, 0.11230469, -0.33984375,
       -0.25390625, 0.05200195, 0.27539062, -0.08398438, -0.31054688,
```

Figure 5.9: Hasil Matrix Shine

- berikut merupakan hasil lpengolahan kata bag pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut. untuk jelasnya dapat di lihat pada gambar 5.10.

```
In [13]: gmodel['bag']
Out[13]:
array([-0.03515625, 0.15234375, -0.12402344, 0.13378906, -0.11328125,
       -0.133667 , -0.16113281, 0.14548438, -0.06835938, 0.140625 ,
       -0.06095859, -0.3046875 , 0.20996694, -0.04345703, -0.2109375 ,
       -0.05957931, -0.05053711, 0.10253906, 0.19842969, -0.09423828,
       0.18847656, -0.07958894, -0.11035156, -0.07910156, 0.06347656,
       -0.15527344, -0.18945312, 0.11132812, 0.27539062, -0.06787109,
       0.01806641, 0.06689453, 0.2578125 , 0.0324707 , -0.24609375,
       -0.05541992, 0.01013184, 0.24121094, -0.21875 , 0.07568359,
       0.09814453, -0.16113281, 0.16503906, -0.09521484, -0.16601562,
       -0.41796875, 0.0300293 , 0.19433594, 0.2890625 , 0.12695312,
       -0.19824219, -0.05517578 , 0.04296875, -0.10107422, 0.07324219,
       -0.13378906, 0.265625 , -0.00466919, 0.19628906, -0.10839844,
       0.14941466, 0.1484375 , 0.05191541, 0.21777348, -0.08544922,
       -0.00195324, 0.02535625 , -0.0319766 , 0.24242193, 0.09628906,
       -0.27539062, -0.09130859 , 0.23730699, 0.08633203, -0.28515625,
       -0.05932617, 0.0011707 , 0.01784421, 0.00955313, -0.12787575 ,
       0.05615234, -0.12207031, -0.09863281, -0.05786133, -0.00375 ,
       -0.30273438, -0.06396484, -0.00744629, -0.17871094, 0.08544922,
       -0.20410156, 0.33789062, 0.00228882, -0.39453125, -0.14453125,
```

Figure 5.10: Hasil Matrix bag

- berikut merupakan hasil lpengolahan kata Car pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut. untuk jelasnya dapat di lihat pada gambar 5.11.
- berikut merupakan hasil lpengolahan kata Wash pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut. untuk jelasnya dapat di lihat pada gambar 5.12.
- berikut merupakan hasil lpengolahan kata Motor pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [14]: gmodel['car']
Out[14]:
array([-0.13085938,  0.00842285,  0.03344727, -0.05883789,  0.04003906,
       -0.14257812,  0.04931641, -0.16894511,  0.20898438,  0.11962691,
       0.18066406, -0.25      , -0.10400391, -0.10742108, -0.01879883,
       0.052080195, -0.00216675,  0.06505312,  0.14453535, -0.04541016,
       0.13161511,  0.16151201, -0.0889579,  0.14453535,  0.0756638,
       0.04467773, -0.15737344, -0.25398625,  0.33984379,  0.09756638,
       -0.25585938, -0.17733389, -0.0329589,  0.16380859, -0.12597656,
       -0.09912109,  0.15893906,  0.06884766, -0.18945312,  0.03823031,
       -0.0534668, -0.03083965,  0.11083984,  0.24121094, -0.234375 ,
       0.12353516, -0.00294465,  0.1484375,  0.33203125,  0.05249923,
       -0.20019531,  0.37695312,  0.12255859,  0.11425781, -0.17675781,
       0.10009766,  0.0039365,  0.26757812,  0.20117188,  0.03710938,
       0.11803984, -0.09814453, -0.3125      ,  0.03515625,  0.02832031,
       0.26171875, -0.08642578, -0.02258301, -0.05834961, -0.00787354,
       0.11767578, -0.04296875, -0.17285156,  0.04394531, -0.23046875,
       0.1640625, -0.11474689, -0.06030273,  0.01196289, -0.24707831,
       0.32617188, -0.04492188, -0.11425781,  0.22851562, -0.01647949,
       -0.15839962, -0.13183594,  0.12597656, -0.17488469,  0.02289473,
       -0.1015625,  0.00817871,  0.16791016, -0.24609375, -0.189375 ,
       -0.09375, -0.01623535, -0.20214844,  0.23144531, -0.05444336,
       -0.05541992, -0.20898438,  0.26757812,  0.27929688,  0.17089644,
       -0.17578125, -0.02770996, -0.20410156,  0.02392578,  0.03125 ,
```

Figure 5.11: Hasil Matrix Car

```
In [15]: gmodel['wash']
Out[15]:
array([ 9.46044922e-03,  1.41601562e-01, -5.46875000e-02,  1.34765625e-01,
       -2.3821250e-01,  3.24218750e-01, -8.44726562e-02, -1.29882812e-01,
       1.07910156e-01,  2.53986250e-01,  1.13525391e-02, -1.66992188e-01,
       -2.79515625e-02,  2.08897812e-01, -4.27246994e-01,  1.05468750e-01,
       -1.40675000e-01,  3.68897500e-01,  2.11328125e-01, -8.00000000e-01,
       -1.6770125e-01, -1.12890625e-01, -1.74569474e-02, -0.02941893e-03,
       6.29862812e-02,  1.62109375e-01,  1.93359375e-01,  2.17285156e-02,
       -2.67028890e-03, -9.13085938e-02, -2.38281250e-01,  2.33632812e-01,
       -8.00781250e-02, -3.80859375e-02, -1.00097566e-01, -1.39648438e-01,
       1.74984688e-01,  6.787110938e-02,  1.11328125e-01,  1.65839062e-01,
       -1.05468750e-01,  2.30712891e-02,  2.00195312e-01, -6.03027344e-02,
       -3.43750000e-01, -1.02050781e-01, -3.88859375e-01, -5.05371094e-02,
       5.07812500e-02,  1.45507812e-01,  2.81250900e-01,  7.03125000e-02,
       2.84423828e-02, -2.29492188e-01, -5.81054688e-02,  4.51668156e-02,
       -3.56445312e-02,  1.77734375e-01,  1.22070312e-01,  3.71093750e-02,
       -1.10839844e-01,  6.83593750e-02, -2.52685547e-02, -1.27292688e-01,
       4.21875000e-01,  5.32226562e-02, -3.92578125e-01,  1.74884688e-01,
       1.77000000e-02, -2.0598125e-02,  2.28125000e-01,  3.00359375e-01,
       1.04089458e-02, -4.05294825e-02, -2.15117388e-01, -2.80896438e-01,
       3.58086710e-02,  8.30078125e-02, -1.6894512e-01,  2.79541816e-02,
       1.04988469e-01, -3.47656250e-01,  5.20019531e-02,  2.24608375e-01,
       1.09677734e-02,  1.69921875e-01, -1.46484375e-01,  2.65625000e-01,
```

Figure 5.12: Hasil Matrix Wash

untuk jelasnya dapat di lihat pada gambar 5.13.

```
In [16]: gmodel['motor']
Out[16]:
array([ 5.73730469e-02,  1.50390625e-01, -4.61425781e-02, -1.32812500e-01,
       -2.59765625e-01, -1.77734375e-01,  3.68652344e-02, -4.37500000e-01,
       2.34375000e-02,  2.57812500e-01,  1.74804688e-01,  2.44148625e-02,
       -2.51953125e-01, -5.76171875e-02,  8.15429688e-02,  1.86767578e-02,
       -3.8338001e-02,  1.58804688e-01, -5.80000000e-01,  1.21914862e-01,
       1.56250600e-01, -4.46804688e-02, -1.32012500e-01, -5.50934375e-01,
       1.20000000e-01, -6.02018750e-01, -1.52573238e-01, -5.50934375e-01,
       3.02734375e-01,  1.53320312e-01, -1.69921875e-01, -1.01074219e-01,
       -3.26538086e-03,  2.28515625e-01,  8.98437500e-02, -7.12896250e-02,
       1.54296875e-01, -8.88671875e-02, -2.36328125e-01,  5.61523438e-03,
       -4.46777344e-02, -3.066460625e-01,  7.42187500e-02,
       -1.30859375e-01,  1.00585938e-01, -3.34472656e-02,  2.10937500e-01,
       3.10058594e-02, -6.50824414e-03,  6.34765625e-02,  4.02832031e-02,
       -2.78320312e-02,  1.07421875e-02,  1.47469388e-01,  2.80761719e-02,
       -1.19390625e-01, -1.37695312e-01,  9.608039738e-02,  1.28906250e-01,
       -3.34472656e-02, -1.08032227e-02, -2.14843750e-01, -9.52148438e-02,
       -6.39648438e-01,  7.51593125e-02, -3.066460625e-01,  2.17774348e-01,
       -2.21679688e-01,  2.33396438e-01,  5.05371094e-02, -3.378960625e-01,
       1.53320312e-01, -7.12896250e-02, -3.68652344e-02,  7.66601562e-02,
       -8.00781250e-02, -1.14257812e-01, -9.71679688e-02, -2.61710750e-01,
       3.84765625e-01, -1.87500000e-01, -1.103515362e-01,  1.00585938e-01,
```

Figure 5.13: Hasil Matrix Motor

- berikut merupakan hasil lpengolahan kata Cycle pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut. untuk jelasnya dapat di lihat pada gambar 5.14.
- berikut merupakan hasil lpengolahan kata love pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut. untuk jelasnya dapat di lihat pada gambar 5.15.
- berikut merupakan hasil lpengolahan kata faith pada data google yang di

```
In [17]: gmodel['cycle']
Out[17]:
array([-0.04541016,  0.21679688, -0.02709961,  0.12353516, -0.20703125,
       -0.1328125,  0.25367188, -0.12389625, -0.125
       -0.18261719, -0.15820312, -0.06176758,  0.21972656, -0.15820312,
       0.02563577, -0.075068359, -0.0625
       0.0464258, -0.31054688,
       -0.03198601,  0.12389625, -0.06176758,  0.21972656, -0.15820312,
       0.07236562, -0.06445312,  0.05517576,  0.14941406, -0.13671875,
       0.19302734,  0.02172852, -0.10693359,  0.02490234, -0.10644531,
       -0.05541992, -0.29492188, -0.49839062,  0.06347656, -0.08447266,
       0.17871894,  0.01165771, -0.01695777,  0.13671875, -0.1640625,
       0.11425781,  0.28809781, -0.06073182, -0.07275391,  0.15033962,
       0.18066406, -0.28515625, -0.04052734,  0.01806641,  0.00331116,
       0.00872083,  0.03564453, -0.29882812,  0.09960938, -0.1484375,
       -0.06787109,  0.05957031, -0.05517576, -0.19628906,  0.2265625,
       0.03173828, -0.07080078,  0.1484375, -0.20214844, -0.03393555,
       0.09863281, -0.02038574, -0.08789062, -0.07226562, -0.09423828,
       -0.17889844,  0.1484375,  0.10546875,  0.06445312,  0.01031494,
       -0.02636719, -0.03686523, -0.125
       -0.06787109,  0.14257812,
       0.37109375, -0.15722656,  0.09326172,  0.34960938, -0.00891553,
```

Figure 5.14: Hasil Matrix Cycle

```
In [18]: gmodel['love']
Out[18]:
array([ 0.18302734, -0.15234375,  0.02587891,  0.16503906, -0.16503906,
       0.06689453, -0.29236875, -0.26367188, -0.149625
       0.20117188,
       -0.02624512, -0.08203125, -0.02770996, -0.04394531, -0.23535156,
       0.16992188, -0.12890625,  0.15722656,  0.08756836, -0.06982422,
       -0.03857422,  0.07958984,  0.22499219, -0.14355469,  0.16796875,
       -0.03515625,  0.05517576,  0.10693359,  0.11181641, -0.16308594,
       -0.11181641,  0.13964844,  0.01556396,  0.12792969,  0.15429688,
       0.07714844,  0.26171875,  0.08642578, -0.02514548,  0.33398438,
       0.18652344, -0.26996094,  0.07080078,  0.02600098, -0.10644531,
       -0.10253906,  0.12384688,  0.04711914,  0.02209473,  0.05834961,
       -0.10986328,  0.14941406, -0.10693359,  0.01556396,  0.08984375,
       0.11230469, -0.04379117, -0.11376953, -0.0037384, -0.01818848,
       0.21679688, -0.12384688,  0.08642578, -0.02514548,  0.33398438,
       0.08667989, -0.22499219, -0.00320878,  0.43139942, -0.15707931,
       0.28515625, -0.08593734, -0.11181641,  0.0213623, -0.30644062,
       -0.09238516, -0.18945312,  0.01512672,  0.18554688, -0.342375,
       -0.21054688,  0.23585894,  0.08740924, -0.2265625, -0.29492188,
       0.08251953, -0.38476562,  0.25390625,  0.26953125,  0.06298828,
```

Figure 5.15: Hasil Matrix love

load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut. untuk jelasnya dapat di lihat pada gambar 5.16.

```
In [19]: gmodel['faith']
Out[19]:
array([-0.26367188, -0.04150391,  0.1953125,  0.13476562, -0.14648438,
       0.11962891,  0.04345703,  0.10351562,  0.12207031,  0.13476562,
       0.06640625,  0.18945312, -0.16601562,  0.21679688, -0.27148438,
       0.3203125,  0.18449219,  0.36132812, -0.1953125, -0.18164062,
       0.15332031, -0.1839844,  0.10253966, -0.01367188,  0.23144531,
       0.05958031, -0.02949219, -0.00320878,  0.26139942, -0.19382724,
       -0.148125, -0.21384688, -0.01376953,  0.01181641, -0.16308594,
       0.04125977,  0.12011719, -0.17480469, -0.22167969, -0.13476562,
       0.3125,  0.06540625, -0.17657578, -0.01708984, -0.1640625,
       -0.02819824,  0.01257324, -0.09521484, -0.18066406, -0.140625,
       -0.02258301,  0.16308594,  0.13183594, -0.08807812,  0.13085938,
       0.27539062, -0.20685469,  0.10351562, -0.20214844, -0.1875,
       0.16992188,  0.13574219,  0.13769531,  0.16308594, -0.03881836,
       -0.11132812,  0.05688477,  0.12255859,  0.09814453, -0.04956055,
       -0.02331543, -0.04248047, -0.080203125,  0.16015625,  0.04150391,
       -0.16601562, -0.13671875,  0.09619141,  0.32617188,  0.08251953,
       -0.20800781,  0.04199219,  0.05834961, -0.27734375,  0.09130859,
       -0.17382812, -0.22460938,  0.03466797,  0.19824219, -0.08837891,
```

Figure 5.16: Hasil Matrix faith

- berikut merupakan hasil lpengolahan kata Fall pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut. untuk jelasnya dapat di lihat pada gambar 5.17.
- berikut merupakan hasil lpengolahan kata Sick pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut. untuk jelasnya dapat di lihat pada gambar 5.18.
- berikut merupakan hasil dari similaritas kata kata yang di olah menjadi matrix tadi adapun persentase untuk perbandingan setiap katanya yaitu 9 persen untuk kata wash dan clear 7 persen untuk kata bag dan love 48

```
In [20]: gmodel['fall']
Out[20]:
array([-0.04272461,  0.10742188, -0.092277344,  0.16894531, -0.1238125 ,
       -0.18693359,  0.04321209,  0.01684327,  0.14649438,  0.15023862,
      -0.08693406,  0.04492188,  0.0145374,  0.08693406, -0.15923862,
      -0.11093156,  0.01092529, -0.08300781, -0.01892899, -0.1953125 ,
      -0.1815625,  0.13671875,  0.09228516, -0.12189375,  0.126953125 ,
      0.03417969,  0.2109375 ,  0.01975399,  0.125,  0.01544189,
      -0.26953125, -0.09898877, -0.07763672, -0.15527344, -0.03393555,
      0.04199219, -0.29882812, -0.18554688,  0.08496994, -0.02087402,
      0.13574219, -0.22558594,  0.33789062, -0.03564453, -0.10839844,
      -0.19335938,  0.0546875 , -0.04956855,  0.3671875 , -0.03295898,
      0.10205078, -0.15136719, -0.00445557,  0.040803966,  0.27539062,
      -0.06933594,  0.05834961,  0.01422119, -0.01397705, -0.05395508,
      -0.0255127 ,  0.06298828,  0.07080078, -0.07617188,  0.06542969,
      -0.01672363, -0.04711914,  0.19623906,  0.08984375,  0.078125 ,
      0.2189375 ,  0.0612793 ,  0.087879862,  0.19628906,  0.11376953,
      0.06542969,  0.03125 ,  0.12868281,  0.02270508,  0.14550781,
      -0.06225586, -0.37695312, -0.05737365, -0.06596484,  0.08984375,
```

Figure 5.17: Hasil Matrix Fall

```
In [21]: gmodel['sick']
Out[21]:
array([ 1.82617188e-01,  1.49414062e-01, -4.05273438e-02,  1.646025050e-01,
       -2.59765625e-01,  3.22265625e-01,  1.73828125e-01, -1.474608936e-01,
      1.01074219e-01,  5.46875000e-02,  1.66992188e-01, -1.68945312e-01,
      2.12411531e-01,  9.38410156e-02, -1.16992188e-01, -1.259765625e-01,
      1.66015625e-01,  1.79867500e-02, -0.938410156e-01,  2.45117188e-01,
      0.74023438e-02, -2.56347656e-02,  3.41796875e-01,  4.98046075e-02,
      1.78710938e-01, -9.918211289e-04,  8.88671875e-02, -1.95212509e-01,
      1.81640625e-01, -2.65625500e-01, -1.45567812e-01,  1.00585939e-01,
      9.42382812e-02, -3.12509000e-02,  1.98974609e-02, -6.39648438e-02,
      1.18652344e-01,  1.23046875e-01, -6.03027344e-02,  4.68750000e-01,
      9.13085938e-02, -3.12509000e-01,  1.84570312e-01, -1.51367188e-01,
      5.78613281e-02, -1.04980469e-01, -1.68945312e-01, -8.00781250e-02,
      -2.01117875e-01,  1.06201172e-02, -1.29882812e-01, -1.25976562e-01,
      -5.56640625e-02,  3.14453125e-01,  5.61523438e-02, -1.28117188e-01,
      7.12890625e-02,  4.37011719e-02,  2.05878125e-01,  5.71289062e-02,
      8.44726562e-02,  2.15820312e-02, -1.26953125e-01,  8.78906250e-02,
      2.48846875e-01, -6.54296875e-02, -2.02636719e-01,  1.52343750e-01,
      -3.57421875e-01,  3.02124023e-03, -2.08007812e-01, -5.05371094e-02],
```

Figure 5.18: Hasil Matrix Sick

persen untuk kata motor dan car 12 persen untuk kata sick dan faith dan terakhir yaitu 6 persen untuk kata cycle dan shine. untuk jelasnya dapat di lihat pada gambar 5.19.

```
In [22]: gmodel.similarity('wash', 'clear')
Out[22]: 0.09919175457242404
In [23]: gmodel.similarity('bag', 'love')
Out[23]: 0.07536096988992866
In [24]: gmodel.similarity('motor', 'car')
Out[24]: 0.4810172832001571
In [25]: gmodel.similarity('sick', 'faith')
Out[25]: 0.12307321986354715
In [26]: gmodel.similarity('cycle', 'shine')
Out[26]: 0.061617924619837734
```

Figure 5.19: hasil dari lima similaritas

2. pada code berikut merupakan hasil dari running code untuk ekstrak word dimana pada baris ke tiga dimasukan perintah untuk menghapus tag html yang terdapat dalam file tersebut selanjutnya pada baris ke 4 yaitu perintah untuk menghilangkan tanda kutip satu selanjutnya pada baris ke 5 yaitu perintah untuk menghapus tanda baca pada file tersebut dan yang terakhir yaitu perintah untuk menghapus double sepasi atau sepasi berurutan. setelah itu dibuat class bari dari random yang bertujuan untuk mengkocok data yang ada pada file tersebut kemudian class permute sentence tersebut akan digunakan untuk mengolah data selanjutnya. untuk lebih jelasnya dapat dilihat pada gambar 5.20.

```

In [31]: import re
...: def extract_words(sent):
...:     sent = sent.lower()
...:     sent = re.sub(r'<[^>]+>', ' ', sent) #hapus tag html
...:     sent = re.sub(r'(\w)\\'(\w)', ' ', sent) #hapus petik
...:     sent = re.sub(r'\W', ' ', sent) #hapus tanda baca
...:     sent = re.sub(r'\s+', ' ', sent) #hapus spasi yang
berurutan
...:     return sent.split()

In [32]: import random
...: class PermuteSentences(object):
...:     def __init__(self, sents):
...:         self.sents = sents
...:
...:     def __iter__(self):
...:         shuffled = list(self.sents)
...:         random.shuffle(shuffled)
...:         for sent in shuffled:
...:             yield sent

```

Figure 5.20: hasil dari ekstrak_words dan permute Sentence

3. gensim merupakan library untuk memodelkan topik unsuperpise atau memodelkan bahasa dengan model unsuperpise. Sadangkan taget dokumen digunakan untuk TaggetDokumen berarti memasukan dokumen untuk di olah oleh mesin. Kemudian Doc2Vect digunakan untuk membandingkan dokumen apakah isi dari dokumen itu bobotnya sama dengan dokumen yang di sandingkannya. untuk lebih jelasnya dapat dilihat pada gambar 5.21 pada gambar tersebut menunjukan di baris ke satau dilakukan dari librari gensim dokumen mengimport method taggedDocument lalu pada baris kedua dari librari gensim model melakukan import metod Doc2Vec.

```

In [7]: from gensim.models.doc2vec import TaggedDocument
...: from gensim.models import Doc2Vec

```

Figure 5.21: Hasil dari penggunaan library gensim , TaggetDocument dan Doc2Vec

4. cara memasukan data traning file pertama tentukan terlebih dahulu tempat file dokumen tersebut disimpan kemudian import librari os setelah itu buat variabel unsup_sentences yang berisikan array kosong, lalu tentukan file yang akan dimasukan setelah itu lakukan os.listdir pada data yang akan dimasukan kemudian semua data tersebut di inisialisasi menjadi f kemudian nama f tersebut dimasukan ke variabel unsup untuk lebih jelasnya dapat di lihat pada gambar 5.22 berikut. pada codingan tersebut merupakan praktikum untuk memasukan data doc2vec

dan untuk hasilruningnya dapat dilihat pada gambar 5.23 kemudian hasilnya dapat dilihat pada gambar 5.24 dimana akan muncul dirname, fname, sent, unsup_sentences dan word dimana data unsup tersebut yang akan di olah nanti

```

# In[11]: unsupervised training data untuk nomer 4
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]))

# In[12]: data kedua

for dirname in ["txt_sentoken/pos","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)]))

# In[13]: data ketiga

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]))

```

Figure 5.22: Codingan untuk memasukan dokumen pelatihan doc2vec

sebagai data seting dan terdapat 391 kata dalam file tersebut. setelah itu dilakukan running lagi pada code berikutnya sehingga akan muncul hasil seperti gambar 5.25 yang mana coding tersebut berguna untuk menambah data unsup_sentences menjadi 160 sekian ribu seperti pada gambar 5.26. Selanjutnya dirunning code ke tiga yang dapat di lihat pada gambar 5.27 untuk menambahkan data menjadi 175 sekian yang hasil akhirnya dapat di lihat pada gambar 5.28 berikut.

```

In [20]: 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]))

```

Figure 5.23: hasil running codingan ke 1

| 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 5.24: Hasil insert data doc2vec dari codingan pertama

5. kenapa harus dilakukan pengocokan data atau rendomisasi ? hal ini harus dilakukan supaya data lebih gambang untuk di olah dan meningkatkan tingkat

```
In [21]: for dirname in ["txt_sentoken/pos","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, send in enumerate(f):
...:                     words = extract_words(sent)
...:                     unsup_sentences.append(TaggedDocument(words,[ "%s/%s-%d" %
...: (dirname,fname,i)]))
```

Figure 5.25: hasil running codingan ke 2

| Name | Type | Size | Value |
|-----------------|------|--------|---|
| dirname | str | 1 | txt_sentoken/neg |
| fname | str | 1 | cv999_14636.txt |
| i | int | 1 | 24 |
| send | str | 1 | after watching _a_night_at_the_roxbury_ , you'll be left with exactly ... |
| sent | str | 1 | David Bryce's comments nearby are exceptionally well written and infor ... |
| unsup_sentences | list | 164720 | [TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument, Tagge ...] |
| words | list | 391 | ['david', 'bryc', 'comments', 'nearby', 'are', 'exceptionally', 'well' ...] |

Figure 5.26: Hasil insert data doc2vec dari codingan ke dua

```
In [22]: 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]))
```

Figure 5.27: hasil running codingan ke 3

| Name | Type | Size | Value |
|-----------------|------|--------|--|
| dirname | str | 1 | txt_sentoken/neg |
| fname | str | 1 | cv999_14636.txt |
| i | int | 1 | 10604 |
| send | str | 1 | after watching _a_night_at_the_roxbury_ , you'll be left with exactly ... |
| 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 5.28: Hasil insert data doc2vec dari codingan ke tiga

akurasi dari proses pengolahan data Doc2Vec. kemudian harus dilakukan pembersihan data agar memori pc atau laptop yang di gunakan untuk mengolah data menjadi ringan dan menambah peforma dari mesin itu sendiri untuk codinan pertama lakukan terlebih dahulu rendomisasi. dapat di lihat pada gambar 5.29 selanjutnya membuat variabel baru dengan nama mute yang di isi data class random dan data unsup_sentence yang dapat dilihat pada gambar 5.30. kemudian setelah pengolahan data dilakukan pembersihan dengan melakukan code pada gambar 5.31

```
...: import random
...: class PermuteSentences(object):
...:     def __init__(self, sents):
...:         self.sents=sents
...:
...:     def __iter__(self):
...:         shuffled = list(self.sents)
...:         random.shuffle(shuffled)
...:         for sent in shuffled:
...:             yield sent
```

Figure 5.29: Membuat random dan membuat class PermuteSentence

```
In [24]: mute=PermuteSentences(unsup_sentences)
```

Figure 5.30: Membuat variabel mute

```
In [26]: model.delete_temporary_training_data(keep_inference=True)
```

Figure 5.31: Code Pembersihan

6. kenapa model harus di save ? suapaya dalam pengolahan data tidak perlu menjalankan kembali data vektorisasi serta untuk meringankan beban ram. kemudian temporary harus dihapus guna meningkatkan peforma komputer. adapun codenya dapat dilihat pada gambar 5.32 dan untuk hasil simpan datanya dapat di lihat pada gambar 5.33.
7. inver_code digunakan untuk membandingkan data doc2vec yang telah di olah dengan kata yang baru atau data yang ada dalam perintah vector itu sendiri contoh membandingkan kata (i will go home) untuk lebih jelasnya dapat di lihat pada gambar 5.34. kemudian untuk hasil running code tersebut dapat di lihat pada gambar 5.35 pada hasil gambar tersebut terdapat hasil vektor yang rata rata berada pada kisaran 0,2 an yang berarti kata yang dimasukan pada

```
# In[18]: kosongin memori biar lega un no 6
model.delete_temporary_training_data(keep_inference=True)

# In[19]: simpan modelnya biar ga bikin ram lemot un no 6
model.save('haci.d2v')
```

Figure 5.32: Membuat variabel mute

| Local Disk (E) > KULIAH > semester_6 > AI > Buku > Chapter03 | | | |
|--|--------------------|----------------------|--------------|
| Name | Date modified | Type | Size |
| aclImdb | 3/28/2019 4:48 PM | File folder | |
| sentiment labelled sentences | 7/5/2016 11:32 AM | File folder | |
| stanfordSentimentTreebank | 3/29/2019 5:45 PM | File folder | |
| txt_sentoken | 3/29/2019 5:42 PM | File folder | |
| forest.csv | 3/22/2019 3:41 PM | Microsoft Excel C... | 73 KB |
| GoogleNews-vectors-negative300.bin | 12/12/2013 7:12 AM | BIN File | 3,558,847 KB |
| haci.d2v | 3/29/2019 11:44 PM | D2V File | 132,049 KB |
| motori.txt | 3/31/2019 11:47 AM | Text Document | 1 KB |

Figure 5.33: Hasil save data vektorisasi

inter_vec datanya ada pada doc2vec atau ada data yang bobotnya menyamai kata-kata di dalam dokumen tersebut.

```
# In[20]: infer nomer 7
model.infer_vector(extract_words("I will go home"))
```

Figure 5.34: Code untuk inver_code

```
In [28]: model.infer_vector(extract_words("I will go home"))
Out[28]:
array([-0.20829524, -0.19678585,  0.28074503,  0.29673776,  0.03135013,
       0.1201962 , -0.01316267,  0.07002044, -0.17286344, -0.10009877,
      -0.11464831, -0.271175 , -0.18410183,  0.13516955, -0.31416982,
     -0.46333492,  0.01430221,  0.06915674, -0.4388135 , -0.01727306,
      0.04100704, -0.09303451, -0.0231339 ,  0.37067318, -0.32521665,
      0.37282658,  0.03136728,  0.09879171,  0.3961481 , -0.36511996,
      0.19673616,  0.19295342,  0.09498549,  0.14677145, -0.18922158,
      0.1511013 ,  0.08283903, -0.25900814, -0.07188511, -0.23026602,
      0.21022144, -0.2707059 , -0.29784605,  0.2009623 , -0.21609542,
     -0.20541847,  0.35160425, -0.31897664,  0.1266007 , -0.11872206,
     -0.01241263, -0.13433872], dtype=float32)
```

Figure 5.35: Hasil dari inver code

8. consine_simirarity setelah melakukan pengolahan data doc2vec dilakukan consine simirarity yang bertujuan untuk membandingkan data berisikan bahasa inggris dengan data yang telah di olah tadi apakah hasilnya mirip atau tidak

untuk caranya yaitu dengan cara mencobacodingan yang terdapat pada gambar 5.36 berikut maka akan muncul hasilnya berapa persen dengan tulisan 0.4 sekian yang berarti tingkat kemiripan dokumen yang di uji tadi untuk hasilnya dapat dilihat pada gambar 5.37 berikut.

```
# In[21]: Mengecek similaritas untuk nomer 8
from sklearn.metrics.pairwise import cosine_similarity
cosine_similirity(
    [model.infer_vector(extract_words("she going to school, after wash hand"))],
    [model.infer_vector(extract_words("Services sucks."))])

# In[22]: Mengecek similaritas 2
from sklearn.metrics.pairwise import cosine_similarity
cosine_similirity(
    [model.infer_vector(extract_words("Dia pergi ke sekolah tadi siang"))],
    [model.infer_vector(extract_words("Services sucks2."))])
```

Figure 5.36: Code untuk consine_simirarity

```
In [29]: from sklearn.metrics.pairwise import cosine_similarity
...: cosine_similirity(
...:     [model.infer_vector(extract_words("she going to school, after wash
hand"))],
...:     [model.infer_vector(extract_words("Services sucks.))])
Out[29]: array([[0.5940237]], dtype=float32)

In [30]: from sklearn.metrics.pairwise import cosine_similarity
...: cosine_similirity(
...:     [model.infer_vector(extract_words("Dia pergi ke sekolah tadi siang"))],
...:     [model.infer_vector(extract_words("Services sucks2."))])
Out[30]: array([[0.6426797]], dtype=float32)
```

Figure 5.37: Hasil dariconsine_simirarity

9. untuk melakukan cross validation pertama masukan terlebih dahulu metode KNeighborsClasifier dan RandomForestClasifier dari library sklearn kemudian dilakukan cross validation setelah itu buat variabel clf dengan isi KNeighborsClasifier dan variabel clfrf dengan isi RandomForestClasifier kemudian di buat skor menggunakan cross validation dengan menggunakan variabel clf dan data sentvecs dan sentiments kemudian dengan numpy dibuat mean dari scores begitu pula untuk variabel clfrf selanjutnya melakukan import metode make_pipeline yang dilakukan untuk membuat skor dari vektorisasi tfidf dan rf. untuk lebih jelasnya dapat di lihat pada gambar 5.38 maka akan muncul hasil rata-rata 0,76 sekian atau 76 persen untuk clf yang dapat dilihat pada gambar 5.39 dan untuk hasil clfrf menghasilkan hasil rata-rata di 71 persen yang dapat dilihat pada gambar 5.40 dan untuk hasil rata-rata keseluruhan cros validation sebesar 0,74 atau 74 persen yang dapat dilihat pada gambar 5.39.

```

# In[25]: instansiasi KNN dan RF nomer 9
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
import numpy as np

clf = KNeighborsClassifier(n_neighbors=9)
clfrf = RandomForestClassifier()

# In[26]: cek score KNN
scores = cross_val_score(clf, sentvecs, sentiments, cv=5)
np.mean(scores), np.std(scores)

# In[27]: cek score RF
scores = cross_val_score(clfrf, sentvecs, sentiments, cv=5)
np.mean(scores), np.std(scores)

# In[28]: lakukan skoring dari vektorisasi, tfidf, dan rf lalu dibuat perbandingan
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)

```

Figure 5.38: Code untuk Cros Validation

```

In [33]: from sklearn.neighbors import KNeighborsClassifier
...: from sklearn.ensemble import RandomForestClassifier
...: from sklearn.model_selection import cross_val_score
...: import numpy as np
...:
...: clf = KNeighborsClassifier(n_neighbors=9)
...: clfrf = RandomForestClassifier()

In [34]: scores = cross_val_score(clf, sentvecs, sentiments, cv=5)
...: np.mean(scores), np.std(scores)
Out[34]: (0.7676666666666667, 0.010780641085864164)

```

Figure 5.39: Hasil dari perhitungan KNeighborsClasifier

```

C:\Users\COKRO\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The default value of n_estimators will
change from 10 in version 0.20 to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\COKRO\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The default value of n_estimators will
change from 10 in version 0.20 to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\COKRO\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The default value of n_estimators will
change from 10 in version 0.20 to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
Out[35]: (0.713, 0.01872016144279863)

```

Figure 5.40: Hasil dari perhitungan RandomForestClasifier

```

C:\Users\COKRO\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The default value of n_estimators will
change from 10 in version 0.20 to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\COKRO\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The default value of n_estimators will
change from 10 in version 0.20 to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\COKRO\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The default value of n_estimators will
change from 10 in version 0.20 to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
Out[36]: (0.742, 0.012355835328567183)

```

Figure 5.41: Hasil dari perhitungan Cross Validation 1

| | | | |
|---------|---------|------|--|
| scores | float64 | (5,) | [0.75 0.76166667 0.72666667 0.735 0.73666667] |
| scores2 | float64 | (5,) | [0.73833333 0.76166667 0.68166667 0.70166667 0.76166667] |

Figure 5.42: Hasil dari perhitungan Cross Validation 2

5.1.3 Penanganan Error

5.2 Fathi Rabbani / 1164074

5.2.1 Teori

1. Why words need to be Vectorizer

karena kata - kata yang digunakan untuk memproses data agar dapat menjadi bagian dari kumpulan data atau atribut yang dapat dibaca oleh sistem machine learning karena sistem tersebut tidak dapat memproses data text secara langsung dan harus di convert terlebih dahulu kedalam bilangan. Untuk ilustrasinya dapat dilihat pada gambar 6.10

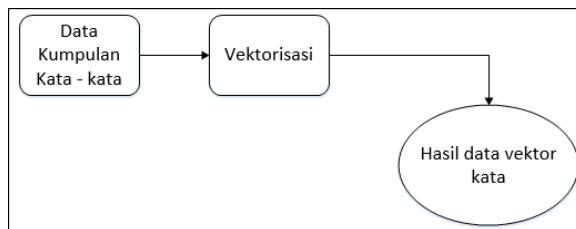


Figure 5.43: Ilustrasi Vektorisasi Kata

2. Why Dimension of Google dataset can reach 300

Dimensi dataset dari google bisa mencapai 300 karena dimensi dari vektor tersebut digunakan untuk membandingkan bobot dari setiap data kata yang diproses. ilustrasi dapat dilihat pada gambar 6.11

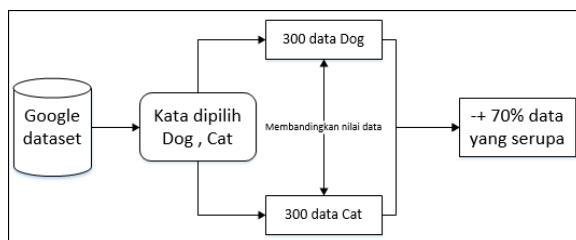


Figure 5.44: Ilustrasi Google dataset

3. Concept of Vectorizer on words

pada vektorisasi dengan menggunakan Word2Vec memiliki kelebihan yang dapat dibedakan dengan penggunaan bag of words yang biasanya. pada bag of word pemrosesan data tidak dapat menganalisa data yang memiliki makna

sama namun penulisannya berbeda, namun pada penggunaan Word2Vec proses tersebut dapat berjalan dengan lebih mudah contohnya adalah penulisan kata please dengan plz. untuk ilustrasi datanya bisa dilihat dalam gambar 6.12

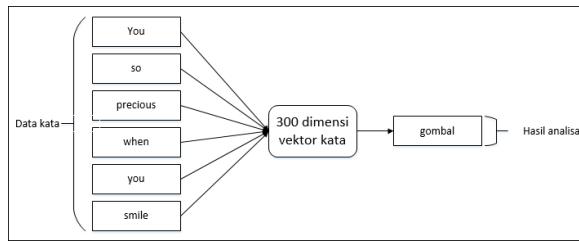


Figure 5.45: Ilustrasi Concept of Vectorizer on Words

4. Concept of Vectorizer on documents

vektorisasi pada Doc2Vec dimana data yang terdapat pada file document tersebut diolah dengan melakukan pemrosesan yang mengutamakan nilai data filenamenya atau atribut utama dimana nilai data inputnya tidak terlalu diproses. ilustrasinya dapat dilihat pada gambar 6.13

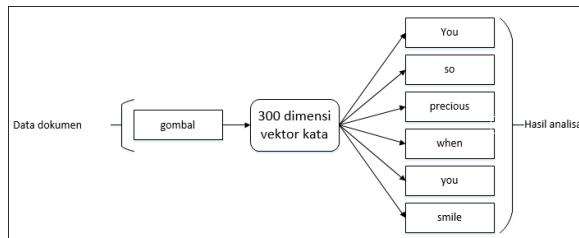


Figure 5.46: Ilustrasi Concept of Vectorizer on Document

5. What is mean and deviation standart

Mean adalah nilai rata-rata dari beberapa buah data. Nilai mean dapat ditentukan dengan membagi jumlah data dengan banyaknya data.

Standar deviasi adalah nilai statistik yang digunakan untuk menentukan bagaimana sebaran data dalam sampel, dan seberapa dekat titik data individu ke mean atau rata-rata nilai sampel.

untuk ilustrasi data mean dan deviation standart bisa dilihat pada gambar 6.14

6. What is skip-gram

| contoh | |
|-----------------|----------|
| nama | ipk |
| fathi | 3 |
| zul | 3.5 |
| puad | 2.8 |
| mean | 3.1 |
| standar deviasi | 0.360555 |

Figure 5.47: Ilustrasi Mean and Deviation Standart

Skip-gram merupakan teknik yang digunakan di area speech processing, dimana n-gram yang dibentuk kemudian ditambahkan juga dengan tindakan skip pada token-tokennya. contohnya terdapat pada gambar 6.15

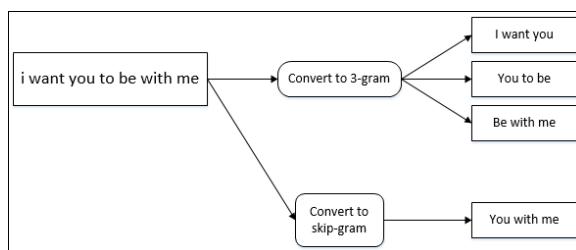


Figure 5.48: Ilustrasi skip-gram

5.2.2 Praktikum

1. Try datasets GoogleNews-vectors

- berikut adalah hasil dari code yang digunakan untuk memanggil data library GENSIM dengan menggunakan perintah import, lalu dari library tersebut diambilah data yang akan digunakan untuk memproses data dari GoogleNews-vector. ilustrasi dapat dilihat pada gambar 6.16
- lalu pada penggunaan code berikut ini akan mengolah data LOVE yang

```
In [6]: import gensim
In [7]: genmod = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=True)
```

Figure 5.49: Ilustrasi import gensim dan olah data GoogleNews-vector

terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar 6.17

```
In [8]: genmod['love']
Out[8]:
array([ 0.10302734, -0.15234375,  0.02587891,  0.16503906, -0.16503906,
       0.06689453,  0.29296875, -0.26367188, -0.140625,  0.20117188,
      -0.02624512, -0.08203125, -0.02770996, -0.04394531, -0.23535156,
      0.16992188,  0.12890625,  0.15722656,  0.00756836, -0.06982422,
     -0.03857422,  0.07958984,  0.22949219, -0.14355469,  0.16796875,
     -0.03515625,  0.05517578,  0.10693359,  0.11181641, -0.16308594,
     -0.11181641,  0.13964844,  0.01556396,  0.12792969,  0.15429688,
     0.07714844,  0.26171875,  0.08642578, -0.02514648,  0.33398438,
     0.18652344, -0.20996094,  0.07080078,  0.02600098, -0.10644531,
    -0.10253906,  0.12304688,  0.04711914,  0.02209473,  0.05834961,
```

Figure 5.50: Ilustrasi hasil olah data LOVE pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data FAITH yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar 6.17

```
In [9]: genmod['faith']
Out[9]:
array([ 0.26367188, -0.04150391,  0.1953125,  0.13476562, -0.14648438,
       0.11962891,  0.04345703,  0.10351562,  0.12207031,  0.13476562,
      0.06640625,  0.18945312, -0.16601562,  0.21679688, -0.27148438,
      0.3203125,  0.10449219,  0.36132812, -0.1953125, -0.18164062,
      0.15332031, -0.10839844,  0.10253906, -0.01367188,  0.23144531,
     -0.05957031, -0.22949219, -0.00604248,  0.26171875,  0.10302734,
     -0.1328125,  0.21484375,  0.01135254,  0.02111816,  0.18554688,
     0.04125977,  0.12011719,  0.17480469, -0.22167969, -0.13476562,
      0.3125,  0.06640625, -0.17675781, -0.01708984, -0.1640625,
     -0.02819824,  0.01257324, -0.09521484, -0.18066406, -0.140625,
```

Figure 5.51: Ilustrasi hasil olah data FAITH pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data FALL yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar 6.18

```
In [10]: genmod['fall']
Out[10]:
array([-0.04272461,  0.10742188, -0.09277344,  0.16894531, -0.1328125,
       -0.10693359,  0.04321289,  0.01904297,  0.14648438,  0.15839062,
      -0.08691406,  0.04492188,  0.0145874,  0.08691406, -0.19824219,
     -0.11035156,  0.01092529, -0.08300781, -0.0189209, -0.1953125,
     -0.1015625,  0.13671875,  0.09228516, -0.12109375,  0.12695312,
     0.03417969,  0.2109375,  0.01977539,  0.125,  0.01544189,
     0.26953125, -0.0098877, -0.07763672, -0.15527344, -0.03393555,
     0.04199219, -0.29882812, -0.18554688,  0.08496094, -0.02087402,
     0.13574219, -0.22558594,  0.33789062, -0.03564453, -0.10839844,
```

Figure 5.52: Ilustrasi hasil olah data FALL pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data SICK yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar 5.53

```
In [11]: genmod ['sick']
Out[11]:
array([- 1.82617188e-01,  1.49414062e-01, - 4.05273438e-02,  1.64062500e-01,
       - 2.59765625e-01,  3.22265625e-01,  1.73828125e-01, - 1.47460938e-01,
       1.01874219e-01,  5.46875000e-02,  1.66992188e-01, - 1.68945312e-01,
      2.24304199e-03,  9.66796875e-02, - 1.66015625e-01, - 1.12304688e-01,
      1.66015625e-01,  1.79687500e-01,  5.92041016e-03,  2.45117188e-01,
     8.74023438e-02, - 2.56347656e-02,  3.41796875e-01,  4.98046875e-02,
    1.78710938e-01, - 9.91821289e-04,  8.88671875e-02, - 1.95312500e-01,
   1.81640625e-01, - 2.65625000e-01, - 1.45507812e-01,  1.008585938e-01,
   9.42382812e-02, - 3.12500000e-02,  1.98974609e-02, - 6.39648438e-02,
```

Figure 5.53: Ilustrasi hasil olah data SICK pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data CLEAR yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar 5.54

```
In [12]: genmod ['clear']
Out[12]:
array([- 2.44140625e-04, - 1.02050781e-01, - 1.49414062e-01, - 4.24804688e-02,
       - 1.67968750e-01, - 1.46484375e-01,  1.76757812e-01,  1.46484375e-01,
       2.26562500e-01,  9.76562500e-02, - 2.67578125e-01, - 1.29882812e-01,
      1.24511719e-01,  2.23632812e-01, - 2.13867188e-01,  3.10858594e-02,
     2.00195312e-01, - 4.76074219e-02, - 6.83593750e-02, - 1.21093750e-01,
      3.22265625e-02,  3.14453125e-01, - 1.11816406e-01,  8.00781250e-02,
     - 2.75878906e-02, - 6.04248407e-03, - 7.37304688e-02, - 1.72851562e-01,
      9.66796875e-02, - 4.91333008e-03, - 1.78710938e-01, - 1.40380859e-03,
     7.91015625e-02,  1.07910156e-01, - 1.10351562e-01, - 8.34960938e-02,
```

Figure 5.54: Ilustrasi hasil olah data CLEAR pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data SHINE yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar 5.55

```
In [13]: genmod ['shine']
Out[13]:
array([- 0.12402344,  0.25976562, - 0.15917969, - 0.27734375,  0.30273438,
       0.09960938,  0.39257812, - 0.22949219, - 0.18359375,  0.3671875 ,
       - 0.10302734,  0.13671875,  0.25390625,  0.07128906,  0.02539062,
      0.21777344,  0.24023438,  0.5234375 ,  0.12304688, - 0.19335938,
     - 0.05883789,  0.0612793 , - 0.01940918,  0.07617188,  0.05102539,
     0.20019531,  0.38085938,  0.00162506, - 0.05029297,  0.14648438,
     - 0.34765625,  0.02563477, - 0.23925781, - 0.04516602, - 0.00479126,
     - 0.24121094, - 0.18945312, - 0.15234375, - 0.05493164,  0.01434326,
     0.390625 , - 0.2109375 ,  0.1484375 , - 0.13183594,  0.24511719,
```

Figure 5.55: Ilustrasi hasil olah data SHINE pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data BAG yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar 5.56

```
In [14]: genmod ['bag']
Out[14]:
array([- 0.03515625,  0.15234375, - 0.12402344,  0.13378906, - 0.11328125,
       - 0.0133667 , - 0.16113281,  0.14648438, - 0.06835938,  0.140625 ,
       - 0.06005859, - 0.3046875 ,  0.20996694, - 0.04345703, - 0.2109375 ,
       - 0.05957031, - 0.05053711,  0.10253906,  0.19042969, - 0.09423828,
      0.18847656, - 0.07958984, - 0.11035156, - 0.07910156,  0.06347656,
     - 0.15527344, - 0.18945312,  0.11132812,  0.27539062, - 0.06787109,
     0.01086641,  0.06689453,  0.2578125 ,  0.0324707 , - 0.24609375,
     - 0.05541992,  0.01013184,  0.24121094, - 0.21875 ,  0.07568359,
     - 0.09814453, - 0.16113281,  0.16503906, - 0.09521484, - 0.16601562,
```

Figure 5.56: Ilustrasi hasil olah data BAG pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data CAR yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar 5.57

```
In [15]: genmod ['car']
Out[15]:
array([ 0.13085938,  0.00842285,  0.03344727, -0.05883789,  0.04003906,
       -0.14257812,  0.04931641, -0.16894531,  0.20898438,  0.11962891,
       0.18064406, -0.25 , -0.10400391, -0.10742188, -0.01879883,
       0.05200195, -0.00216675,  0.06445312,  0.14453125, -0.04541016,
       0.16113281, -0.01611328, -0.03088379,  0.08447266,  0.16210938,
       0.04467773, -0.15527344,  0.25390625,  0.33984375,  0.00756836,
       -0.25585938, -0.01733398, -0.03295898,  0.16308594, -0.12597656,
       -0.09912109,  0.16503906,  0.06884766, -0.18945312,  0.02832031,
       -0.0534668 , -0.03063965,  0.11083984,  0.24121094, -0.234375 ,
```

Figure 5.57: Ilustrasi hasil olah data CAR pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data WASH yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar 5.58

```
In [16]: genmod ['wash']
Out[16]:
array([ 9.46044922e-03,  1.41601562e-01, -5.46875000e-02,  1.34765625e-01,
       -2.38281250e-01,  3.24218750e-01, -8.44726562e-02, -1.29882812e-01,
       1.07910156e-01,  2.53906250e-01,  1.13525391e-02, -1.66992188e-01,
       -2.79541016e-02,  2.08007812e-01, -4.27246094e-02,  1.05468750e-01,
       -7.42187500e-02,  3.04687500e-01,  2.11914062e-01, -8.88671875e-02,
       2.67578125e-01,  2.12890625e-01,  1.74560547e-02,  2.02941895e-03,
       6.29882812e-02,  1.62109375e-01,  1.93359375e-01,  2.17285156e-02,
       -2.67028809e-03, -9.13085938e-02, -2.38281250e-01,  2.23632812e-01,
```

Figure 5.58: Ilustrasi hasil olah data WASH pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data MOTOR yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar 5.59

```
In [17]: genmod ['motor']
Out[17]:
array([ 5.73730469e-02,  1.58390625e-01, -4.61425781e-02, -1.32812500e-01,
       -2.59765625e-01, -1.77734375e-01,  3.68652344e-02, -4.37500000e-01,
       2.34375000e-02,  2.57812500e-01,  1.74804688e-01,  2.44140625e-02,
       -2.51953125e-01, -5.76171875e-02,  8.15429688e-02,  1.86767578e-02,
       -3.83300781e-02,  1.58203125e-01, -5.85937500e-02,  1.12304688e-01,
       1.56250000e-01, -4.24804688e-02, -1.32812500e-01,  2.11914062e-01,
       1.23046875e-01,  1.69921875e-01, -1.55273438e-01,  4.58984375e-01,
       3.02734375e-01,  1.53320312e-01, -1.69921875e-01, -1.01874219e-01,
       -3.26538086e-03,  2.28515625e-01,  8.98437500e-02, -7.12890625e-02,
```

Figure 5.59: Ilustrasi hasil olah data MOTOR pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data CYCLE yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar 5.60
- dan pada hasil code berikut ini adalah hasil dari proses penggunaan pertama code similarity yang akan menghitung nilai value data yang dibandingkan dengan masing - masing kata seperti pada hasil dari perbandingan

```
In [18]: genmod['cycle']
Out[18]:
array([ 0.04541016,  0.21679688, -0.02709961,  0.12353516, -0.20703125,
       -0.1328125 ,  0.26367188, -0.12890625, -0.125     ,  0.15332031,
      -0.18261719, -0.15820312, -0.06176758,  0.21972656, -0.15820312,
      0.02563477, -0.07568359, -0.0625     ,  0.04614258, -0.31054688,
     -0.1378906, -0.11669922, -0.3359375 ,  0.078125     ,  0.08447266,
      0.07226562, -0.06445312,  0.05517578,  0.14941406,  0.13671875,
      0.10302734,  0.02172852, -0.10693359,  0.02490234, -0.10644531,
     -0.05541992, -0.29492188, -0.40039062,  0.06347656, -0.08447266,
      0.17871094,  0.01165771, -0.01696777,  0.13671875, -0.1640625 ,
```

Figure 5.60: Ilustrasi hasil olah data CYCLE pada GoogleNews-vector

kata LOVE disandingkan dengan FAITH menghasilkan nilai 37 persen, sedangkan kata WASH dan SHINE menghasilkan nilai 27 persen dan kata CAR yang disandingkan dengan kata MOTOR menghasilkan 48 persen, dimana kita dapat menyimpulkan bahwa semakin data kata tersebut memiliki tingkat kesamaan yang tinggi maka nilai hasil yang ditampilkan pun akan semakin tinggi. ilustrasi bisa dilihat pada gambar 5.61

```
In [19]: genmod.similarity('love', 'faith')
Out[19]: 0.3705347934587281

In [20]: genmod.similarity('wash', 'shine')
Out[20]: 0.2770128965426825

In [21]: genmod.similarity('car', 'motor')
Out[21]: 0.4810172832001571

In [22]: genmod.similarity('bag', 'cycle')
Out[22]: 0.040672609213443504

In [23]: genmod.similarity('shine', 'fall')
Out[23]: 0.27789493775772145
```

Figure 5.61: Ilustrasi hasil olah data pada GoogleNews-vector menggunakan SIMILARITY

2. extract_words dan PermutatedSentences

pada penjelasan berikut ini akan menyangkut pembersihan data yang akan digunakan untuk diproses, dimana data akan di EXTRACT dari setiap katanya agar terbebas dari data TAG HTML, APOSTROPES, TANDA BACA, dan SPASI yang berlebih. dengan menggunakan perintah code STRIP dan SPLIT. lalu penggunaan library random yang akan dibuat untuk melakukan KOCLOK data dengan acuan datanya adalah data yang terdapat pada variable KATA. untuk ilustrasi hasil dari codenya dapat dilihat pada gambar 5.62

3. TaggedDocument dan Doc2Vec

```
In [28]: import re
...: def extract_words(kata):
...:     kata = kata.lower()
...:     kata = re.sub(r'<[^>]+>', ' ', kata)
...:     kata = re.sub(r'(\w)\' (\w)', '\1\2', kata)
...:     kata = re.sub(r'\w', ' ', kata)
...:     kata = re.sub(r'\s+', ' ', kata)
...:     kata = kata.strip()
...:     return kata.split()
...:

...: import random
...: class PermuteSentences(object):
...:     def __init__(self, lenght):
...:         self.lenght = lenght
...:
...:     def __iter__(self):
...:         req = list(self.lenght)
...:         random.shuffle(req)
...:         for kata in req:
...:             yield kata
```

Figure 5.62: Ilustrasi hasil olah data pada GoogleNews-vector menggunakan extract_words dan PermuteSentences

gensim merupakan open-source model ruang vektor dan toolkit topic modeling, yang diimplementasikan dalam bahasa pemrograman Python. Untuk kinerja Gensim, digunakan NumPy, SciPy dan Cython (opsional). Gensim secara khusus ditujukan untuk menangani koleksi teks besar dengan menggunakan algoritma secara online. Gensim mengimplementasikan tf-idf, latent semantic analysis (LSA), Latent Dirichlet Analysis (LDA), dan lain-lain.

tagged document merupakan sebuah class yang terdapat pada pemrosesan data pada library gensim yang akan mengolah data teks yang ada pada dokumen - dokumen yang dipakai.

Doc2Vec merupakan algoritma doc embedding, yaitu pemetaan dari dokumen menjadi vektor, serta pemetaan data dokumen 1 dan dokumen lainnya. ilustrasi dari tagged document dan Word2Vec ada pada gambar 5.63

```
In [2]: from gensim.models.doc2vec import TaggedDocument
...: from gensim.models import Doc2Vec
```

Figure 5.63: Ilustrasi TaggedDocument dan Doc2Vec

4. Praktek data training

pertama buka data training yang akan diolah pada aplikasi python, import library OS dan membuat data variable unsup_sentences dengan nilai array kosong. buatkan data direktori untuk memanggil data yang akan diolah dan

buatkan juga variable data nilai fname yang akan memproses data dirname untuk diisikan pada variable unsup_sentences. code yang digunakan dapat dilihat pada gambar 5.64

```
# In[21]
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:
                kata = f.read()
                words = extract_words(kata)
                unsup_sentences.append(TaggedDocument(words, [dirname + "/" + fname]))

# In[22]
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, kata in enumerate(f):
                    words = extract_words(kata)
                    unsup_sentences.append(TaggedDocument(words, ["%s/%s-%d" % (dirname, fname, i)]))

# In[23]
with open("StanfordSentimentTreebank/original_rt_snippets.txt", encoding='UTF-8') as f:
    for i, line in enumerate(f):
        words = extract_words(line)
        unsup_sentences.append(TaggedDocument(words, ["rt-%d" % i]))
```

Figure 5.64: Ilustrasi data code praktek data training

data pada hasil code digambar berikut 5.65, menghasilkan data pada gambar 5.66 yang akan memunculkan data variable DIRNAME, FNAME, KATA dan unsup_sentences yang memiliki data sebanyak 55 kata dalam file yang diolah tersebut. hasil run dengan menggunakan code pada gambar 5.67, menghasilkan data nilai yang terdapat pada gambar 5.68. lalu pada code yang terdapat digambar 5.69, menghasilkan data 5.70.

```
# In[21]
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:
                kata = f.read()
                words = extract_words(kata)
                unsup_sentences.append(TaggedDocument(words, [dirname + "/" + fname]))
```

Figure 5.65: Ilustrasi data code praktek data training

| Name | Type | Size | Value |
|-----------------|------|-------|---|
| dirname | str | 1 | test/neg |
| fname | str | 1 | 9_4.txt |
| kata | str | 1 | David Bryce's comments nearby are exceptionally well written and infor ... |
| unsup_sentences | list | 73293 | [TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument, ...] |
| words | list | 55 | [', ., , , ., !, ., "", "...", "", "...", .., ...] |

Figure 5.66: Ilustrasi data code praktek data training

5. Why need Shuffled and Clean memory

dilakukan shuffled adalah agar datanya lebih mudah untuk diolah dan untuk menentukan tingkat tinggi akurasi dari hasil pemrosesan. dan dilakukan

```
# In[22]
for dirname in ["review_polarity/txt_sentoken/pos", "review_polarity/txt_sentoken/neg"]:
    for fname in sorted(os.listdir(dirname)):
        if fname[:2] == "cv":
            with open(dirname + "/" + fname, encoding='UTF-8') as f:
                for i, kata in enumerate(f):
                    words = extract_words(kata)
                    unsup_sentences.append(TaggedDocument(words, ["%s/%s-%d" % (dirname, fname, i)]))
```

Figure 5.67: Ilustrasi data code praktek data training

| Name | Type | Size | Value |
|-----------------|------|--------|---|
| dirname | str | 1 | review_polarity/txt_sentoken/neg |
| fname | str | 1 | cv999_14636.txt |
| i | int | 1 | 24 |
| kata | str | 1 | after watching _a_night_at_the_roxbury_ , you'll be left with exactly ... |
| unsup_sentences | list | 138013 | [TaggedDocument, TaggedDocument, TaggedDocument, Tagge ...] |
| words | list | 3 | ['.', ' ', '.'] |

Figure 5.68: Ilustrasi data code praktek data training

```
# In[23]
with open("stanfordSentimentTreebank/original_rt_snippets.txt", encoding='UTF-8') as f:
    for i, line in enumerate(f):
        words = extract_words(kata)
        unsup_sentences.append(TaggedDocument(words, ["rt-%d" % i]))
```

Figure 5.69: Ilustrasi data code praktek data training

| Name | Type | Size | Value |
|-----------------|------|--------|--|
| dirname | str | 1 | review_polarity/txt_sentoken/neg |
| fname | str | 1 | cv999_14636.txt |
| i | int | 1 | 10604 |
| kata | str | 1 | after watching _a_night_at_the_roxbury_ , you'll be left with exactly ... |
| line | str | 1 | Her fans walked out muttering words like ``horrible'' and ``terrible,' ... |
| unsup_sentences | list | 148618 | [TaggedDocument, TaggedDocument, TaggedDocument, Tagge ...] |
| words | list | 3 | ['.', ' ', '.'] |

Figure 5.70: Ilustrasi data code praktek data training

pembersihan memory adalah agar chace yang disimpan tidak membuat proses pada komputer menjadi lambat dan dapat digunakan untuk memproses data lainnya agar menjadi lebih ringan dan cepat. pada gambar 5.71 adlah proses untuk melakukan pengoclokan data dan pada gambar 5.72 adalah proses untuk memasukan data unsup_sentences kedalam variable muter untuk diproses dengan class PermuteSentences. dan pada gambar ?? adalah code yang digunakan untuk membersihkan data memory.

```
In [28]: import re
.... def extract_words(kata):
....     kata = kata.lower()
....     kata = re.sub(r'<[^>]+>', ' ', kata)
....     kata = re.sub(r'(\w)\` (\w)', '\1\2', kata)
....     kata = re.sub(r'\w', ' ', kata)
....     kata = re.sub(r'\s+', ' ', kata)
....     kata = kata.strip()
....     return kata.split()
.....
.....
.... import random
.... class PermuteSentences(object):
....     def __init__(self, lenght):
....         self.lenght = lenght
.....
....     def __iter__(self):
....         req = list(self.lenght)
....         random.shuffle(req)
....         for kata in req:
....             yield kata
```

Figure 5.71: Ilustrasi Shuffled dan Randomisasi data

```
In [10]: muter = PermuteSentences(unsup_sentences)
.... mod = Doc2Vec(muter, dm=0, hs=1, size=50)
```

Figure 5.72: Ilustrasi pembuatan variable muter untuk memuat data unsup_sentences

```
mod.delete_temporary_training_data(keep_inference=True)
```

Figure 5.73: Ilustrasi code untuk membersihkan data memory

6. Why model have to be saved

dalam pengolahan data dengan menggunakan proses yang panjang ditakutkan data yang sudah diproses tersebut dapat hilang jika terdapat kejadian atau emergency pada saat pengolahan dan pemrosesan data, misalnya harddisk error atau pun listrik yang padam. dan proses penyimpanan data juga dilakukan agar data yang sudah diolah data dipanggil lagi tanpa harus melakukan proses dari awal sehingga tidak memakan waktu. untuk code yang digunakan dapat dilihat pada gambar 5.74

```
mod.save('simpanan.d2v')
```

Figure 5.74: Ilustrasi data code save data

berikut ini adalah hasil file dari penggunaan code save tersebut. bisa dilihat pada gambar 5.75

| | | | |
|----------------------------|--------------------|----------------------|-----------|
| SentimentAnalysis | 3/18/2019 3:14 PM | Python Source File | 2 KB |
| SentimentAnalysisnew.ipynb | 3/18/2019 3:14 PM | Jupyter Notebook | 90 KB |
| simpanan.d2v | 3/30/2019 1:22 AM | D2V File | 54,642 KB |
| tugas5 | 3/28/2019 11:41 PM | Python Source File | 2 KB |
| worddoc2vec | 3/30/2019 12:05 AM | Python Source File | 6 KB |
| Youtube01-Psy | 3/18/2019 3:14 PM | Microsoft Excel C... | 57 KB |
| Youtube02-KatyPerry | 3/18/2019 3:14 PM | Microsoft Excel C... | 64 KB |
| Youtube03-LMFAO | 3/18/2019 3:14 PM | Microsoft Excel C... | 64 KB |
| Youtube04-Eminem | 3/18/2019 3:14 PM | Microsoft Excel C... | 82 KB |
| Youtube05-Shakira | 3/18/2019 3:14 PM | Microsoft Excel C... | 72 KB |

Figure 5.75: Ilustrasi hasil file simpan

7. infer_vector

berfungsi untuk dokumen baru, dan bisa menggunakan data vektor yang dilatih secara massal, seperti yang disimpan dalam model, untuk dokumen yang merupakan bagian dari data training. untuk percobaannya dapat dilihat pada gambar ??

```
In [13]: mod.infer_vector(extract_words("This Place is not worth your time,  
let alone Vegas."))
Out[13]:
array([-0.00374494, -0.0017316, -0.00647218, -0.00174912, 0.00963611,
       -0.0038244, 0.00033789, -0.000808683, 0.00724112, 0.002089,
       -0.00337453, -0.00712116, -0.00925899, 0.00921614, 0.00114124,
       0.00834063, -0.00195045, -0.00096859, -0.00773017, 0.00428807,
       0.0008305, 0.00776206, -0.00713274, 0.00674286, 0.00414128,
       -0.0092394, -0.00848763, 0.00395087, 0.00693592, -0.00737078,
       -0.00567001, 0.00646778, -0.00077156, -0.00292188, 0.00395201,
       -0.00054506, -0.00203164, -0.00913866, -0.00929867, 0.00768393,
       0.0052638, 0.00927046, -0.00921578, -0.00850527, 0.00733398,
       0.0069659, -0.00279458, -0.00753236, 0.00648478, -0.00494211],
      dtype='float32')
```

Figure 5.76: Ilustrasi code dan hasil infer_vector

8. cosine_similarity

merupakan sebuah algoritma yang digunakan untuk membandingkan dari dua buah data yang bukan merupakan data vector untuk menguji nilai kemiripan data satu dengan data lainnya. hasil dari percobaan pada tugas no 8 ini dapat dilihat pada gambar 5.77 yang menghasilkan nilai akurasi sebesar 20 persen dan gambar 5.78 yang menghasilkan nilai akurasi sebesar 91 persen.

9. Cross Validation

```
In [12]: from sklearn.metrics.pairwise import cosine_similarity
...: cosine_similarity(
...:     [mod.infer_vector(extract_words("Highly recommended."))],
...:     [mod.infer_vector(extract_words("Services sucks."))])
Out[12]: array([[0.206962]], dtype=float32)
```

Figure 5.77: Ilustrasi code dan hasil penggunaan cosine_similarity

```
In [13]: from sklearn.metrics.pairwise import cosine_similarity
...: cosine_similarity(
...:     [mod.infer_vector(extract_words("tolong bantuan."))],
...:     [mod.infer_vector(extract_words("tolong bantuan."))])
Out[13]: array([[0.91143525]], dtype=float32)
```

Figure 5.78: Ilustrasi code dan hasil penggunaan cosine_similarity

pertama melakukan import data dari library KNeighborsClassifier, RandomForestClassifier, cross_val_score dan numpy yang digunakan untuk membuat data cross validasi dapat dilihat pada gambar 5.79.

```
In [16]: from sklearn.neighbors import KNeighborsClassifier
...: from sklearn.ensemble import RandomForestClassifier
...: from sklearn.model_selection import cross_val_score
...: import numpy as np
...:
...: clf = KNeighborsClassifier(n_neighbors=9)
...: clfrf = RandomForestClassifier()
```

Figure 5.79: Ilustrasi memasukan code import library

lalu selanjutnya membuat data variable scores yang akan memuat nilai cross_val_score dengan datanya diambil dari KNeighborsClassifier yang terdiri dari sentvecs, sentiments dan clf dan mengolahnya menggunakan numpy untuk menampilkan data pada gambar 5.80 yang menghasilkan nilai akurasi sebesar 53 persen.

```
In [40]: scores = cross_val_score(clf, sentvecs, sentiments, cv=5)
...: np.mean(scores), np.std(scores)
Out[40]: (0.5283333333333334, 0.006411794687223791)
```

Figure 5.80: Ilustrasi perhitungan data KNeighborsClassifier dengan cross validasi

membuat data variable scores yang akan memuat nilai cross_val_score dengan datanya diambil dari RandomForestClassifier yang terdiri dari sentvecs, sentiments dan clfrf dan mengolahnya menggunakan numpy untuk menampilkan data pada gambar 5.81 yang menghasilkan nilai akurasi sebesar 53 persen.

penggunaan make_pipeline adalah untuk membuat data dari KNeighborsClassifier, RandomForestClassifier dan Vectorizer digabungkan untuk menghasilkan data nilai pada gambar 5.82 menghasilkan nilai akurasi sebesar 74 persen.

dan keseluruhan nilai yang tercatat terdapat pada gambar 5.83.

```
In [41]: scores2 = cross_val_score(clfrf, sentvecs, sentiments, cv=5)
...: np.mean(scores2), np.std(scores2)
C:\Users\Fathi-PC\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Fathi-PC\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Fathi-PC\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Fathi-PC\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Fathi-PC\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
Out[41]: (0.5319999999999999, 0.006446359868604594)
```

Figure 5.81: Ilustrasi perhitungan data RandomForestClassifier dengan cross validasi

```
In [42]: from sklearn.pipeline import make_pipeline
...: from sklearn.feature_extraction.text import CountVectorizer,
TfidfTransformer
...: pipeline = make_pipeline(CountVectorizer(), TfidfTransformer(),
RandomForestClassifier())
...:
...: scores3 = cross_val_score(pipeline, sentences, sentiments, cv=5)
...: np.mean(scores3), np.std(scores3)
C:\Users\Fathi-PC\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Fathi-PC\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Fathi-PC\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Fathi-PC\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Fathi-PC\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Fathi-PC\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
Out[42]: (0.7416666666666667, 0.02345207879911713)
```

Figure 5.82: Ilustrasi perhitungan data Cross Validasi untuk nilai keseluruhan dari KNeighborsClassifier, RandomForestClassifier dan Vectorizer

| | | |
|---------|--------------|--|
| scores | float64 (5,) | [0.53333333 0.555 0.52333333 0.51833333 |
| scores2 | float64 (5,) | [0.53833333 0.52333333 0.52666667 0.53166667 0.54] |
| scores3 | float64 (5,) | [0.74666667 0.77333333 0.71833333 0.71166667 0.75833333] |

Figure 5.83: Ilustrasi perhitungan data hasil persenan yang diakumulasi

Chapter 6

Discussion

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

6.1 Cokro Edi Prawiro / 1164069

6.1.1 Teori

1. Jelaskan kenapa file suara harus dilakukan MFCC dilengkapi dengan ilustrasi atau gambar.

digunakan untuk mengidentifikasi jenis suara misalkan jenis suara gendre lagu jes pop metal dan klasikal atau suara ultra sonic untuk contohnya dapat di gambar 6.1

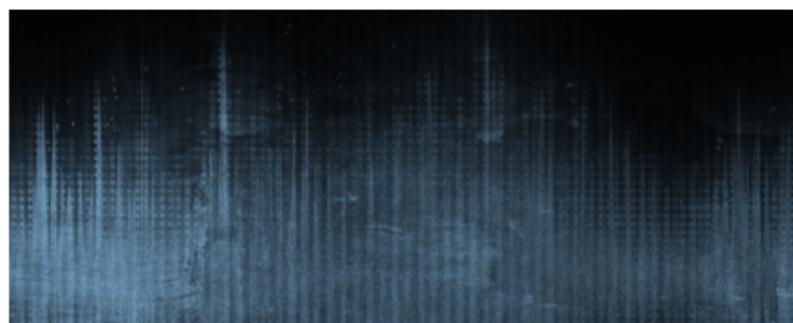


Figure 6.1: Ilustrasi gambar metode MFCC

2. Jelaskan konsep dasar neural network. dilengkapo dengan ilustrasi gambar.

konsep neural network dilaka ada inputan pasti ada outputan sesuai dengan kategori inputan dan fungsi di dalamnya. untuklebih jelasnya dapat dilihat pada ilustrasi gambar berikut. 6.2

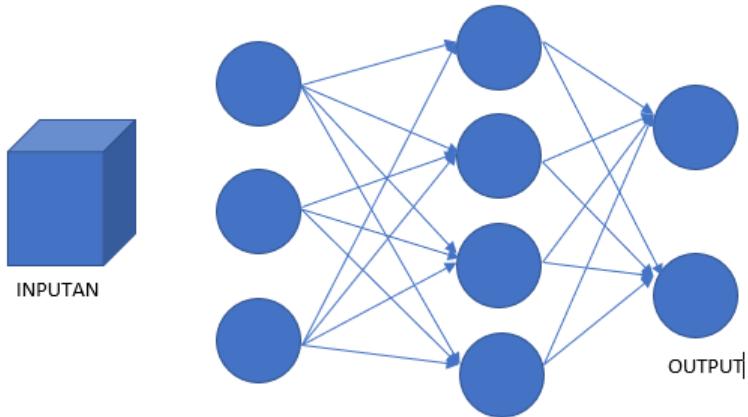


Figure 6.2: Ilustrasi Konsep dasar neural network

3. Jelaskan konsep pembobotan dalam neural network. dilengkapi dengan ilustrasi gambar.

pembobotan dalam neural network yaitu digunakan untuk membedakan objek inputan atau variabel inputan untuk AI misalkan apel dan jeruk digunakan untuk variabel inputan maka dibuat pembobotan antara kedua benda tersebut untuk menentukan output yang pasti dari inputan yang dilakukan untuk lebih jelasnya dapat dilihat pada gambar. 6.3

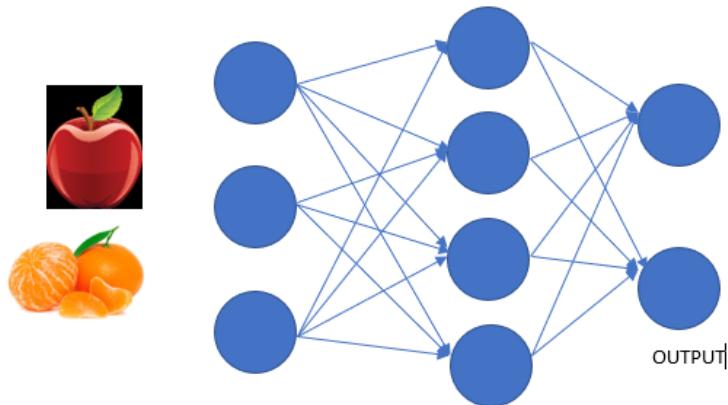


Figure 6.3: Ilustrasi Konsep pembobotan pada neural network

4. Jelaskan konsep aktifitas dalam neural network. dilengkapi dengan ilustrasi gambar.

cara aktifitas dalam neural network dilakukan terhadap input pada neural network inputan tersebut dimasukan kepada fungsi pada mesin misalkan fungsi

$\tanh(x)$ sehingga dihasilkanlah output yang sesuai dengan fungsi tersebut. Untuk lebih jelasnya dapat dilihat pada gambar 6.4

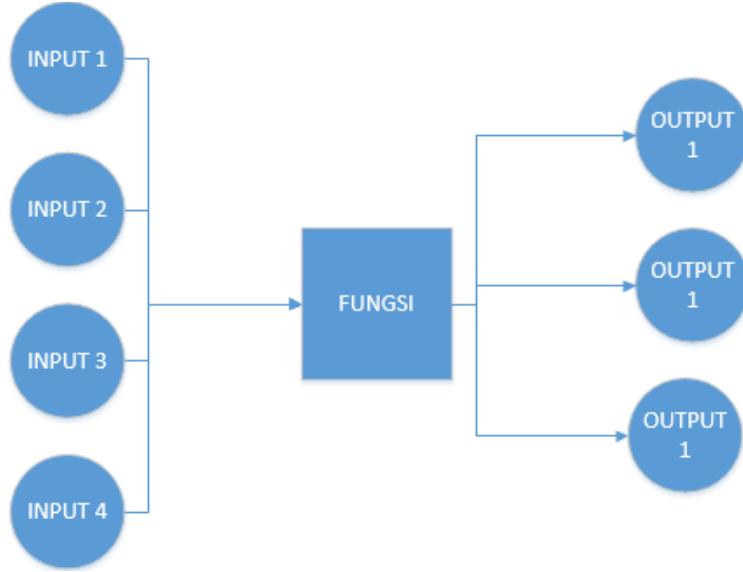


Figure 6.4: Gambar yang dibaca hasil plotnya

5. Jelaskan cara membaca hasil plot dari MFCC dilengkapi dengan ilustrasi gambar.

Cara membaca hasil plotting dari MFCC yaitu tentukan terlebih dahulu batas minimal Hz dari gelombang suara dan batas maksimal dari suara tersebut. kemudian warna yang paling pekat merupakan hasil dari pengolahan data tersebut misalkan muncul warna orange pekat di bagian bawah dan orange muda di bagian atas yang berarti suara tersebut kuat bagian basnya dan biasanya juga antara warna yang pekat tersebut ada jarak. Untuk lebih jelasnya dapat dilihat pada gambar 6.5 berikut.

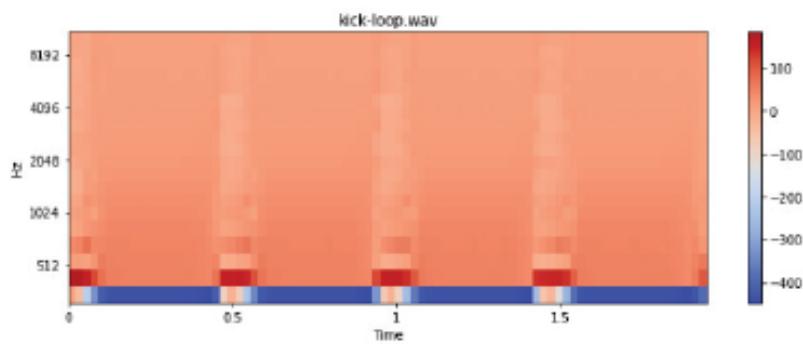


Figure 6.5: Ilustrasi Cara Membaca Hasil Plot

6. Jelaskan apa itu one-hot encoding, dilengkapi dengan ilustrasi kode atau gambar.

one-hot encoding merupakan pemberian nilai pada suatu variabel jika nilai itu iya maka nilainya satu dan jika tidak maka nilainya nol untuk lebih jelasnya dapat dilihat pada gambar 6.6 tersebut

| | pop | rock | blues | rege | clasical |
|--------|-----|------|-------|------|----------|
| Lagu 1 | 1 | 0 | 0 | 0 | 0 |
| Lagu 2 | 1 | 0 | 0 | 0 | 0 |
| Lagu 3 | 0 | 1 | 0 | 0 | 0 |
| Lagu 4 | 0 | 0 | 1 | 0 | 0 |
| Lagu 5 | 0 | 0 | 0 | 1 | 0 |
| Lagu 6 | 0 | 0 | 0 | 0 | 1 |
| Lagu 7 | 0 | 0 | 0 | 0 | 1 |

Figure 6.6: Ilustrasi Konsep one-hot encoding

7. Jelaskan apa dari np.unique dan to_categorical dalam kode program, dilengkapi dengan ilustrasi atau gambar.

digunakan untuk membuat array sedangkan to_categorical digunakan untuk membuat matrix bauititu 64 bit atau 32 bit. Untuk contoh gambarnya dapat dilihat pada gambar 6.7 dan gambar 6.8

```
>>> np.unique([1, 1, 2, 2, 3, 3])
array([1, 2, 3])
>>> a = np.array([[1, 1], [2, 3]])
>>> np.unique(a)
array([1, 2, 3])
```

Figure 6.7: Ilustrasi np.unique

to_categorical

```
keras.utils.to_categorical(y, num_classes=None, dtype='float32')
```

Figure 6.8: Ilustrasi to_categorical

8. Jelaskan apa fungsi dari Sequential dalam kode program, dilengkapi dengan ilustrasi atau gambar.

sequential adalah proses perbandingan setiap elemen satu persatu mulai dari objek pertama hingga yang di tuju atau jika mencari angka 100 maka sequential akan membagi bagian misalnya dari satu sampai 20 dan seterusnya sampai mendapat nilai seratus. untuk lebih jelasnya dapat dilihat gambar 1.51 berikut:

| Bobot 1 | Bobot 2 | Bobot 3 | Bobot 4 | Bobot 5 |
|---------|---------|---------|---------|---------|
| 1-20 | 21-40 | 41-60 | 61-80 | 81-100 |

Figure 6.9: Ilustrasi Konsep pembobotan pada neural network

6.2 Fathi Rabbani / 1164074

6.2.1 Teori

1. Kenapa file suara harus dilakukan MFCC

MFCC digunakan untuk mengidentifikasi jenis suara yang dimasukan seperti suara lagu, hujan atau pun suara yang tidak dapat didengar oleh telinga manusia yaitu Ultrasonik, sehingga dibutuhkan penggunaan MFCC untuk memproses data tersebut agar dapat dibaca oleh manusia. ilustrasi untuk MFCC dapat dilihat pada gambar 6.10

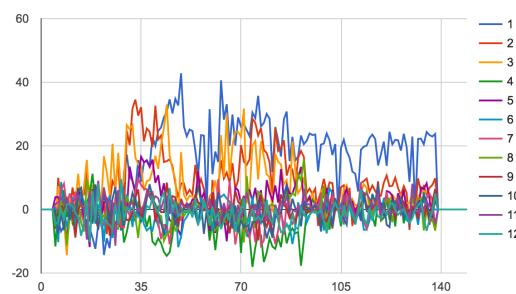


Figure 6.10: Ilustrasi MFCC

2. Konsep dasar Neural Network

Neural Network sebenarnya mengadopsi dari kemampuan otak manusia yang mampu memberikan stimulasi/rangsangan, melakukan proses, dan memberikan output. Output diperoleh dari variasi stimulasi dan proses yang terjadi

di dalam otak manusia. ilustrasi Neural Network dapat dilihat pada gambar 6.11

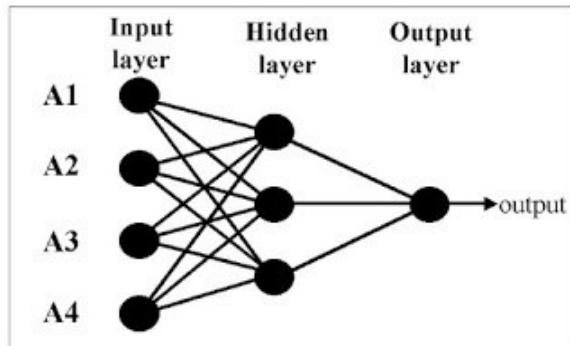


Figure 6.11: Ilustrasi Neural Network

3. Konsep Pembobotan dalam Neural Network

konsep pembobotan dalam neural network digunakan untuk membedakan data satu dengan data lainnya, sebagai contoh dapat dilihat pada gambar 6.12. dimana data inputan yang masuk adalah 2 data yaitu "anjing" dan "kucing" yang diolah dengan proses membandingkan data dan diolah melalui pembobutan sehingga menampilkan hasil output.

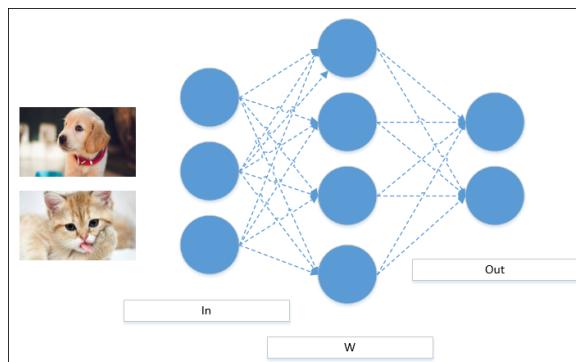


Figure 6.12: Ilustrasi pembobotan Neural Network

4. Konsep Aktifasi dalam Neural Network

dalam Neural Network fungsi aktifasi meliputi inputan dan diproses sehingga menghasilkan Output nilai yang diharapkan, dengan menggunakan persepsi alur cara penyampaian informasi pada jaringan syaraf otak. ilustrasi dapat dilihat pada gambar 6.13

5. cara membaca hasil PLOT dari MFCC

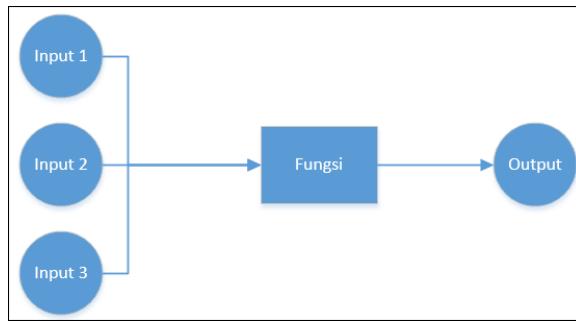


Figure 6.13: Ilustrasi fungsi aktifasi Neural Network

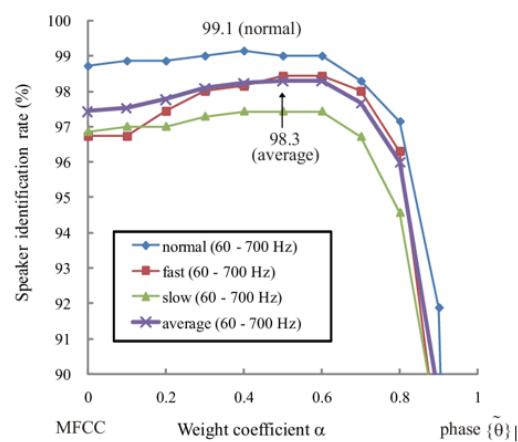


Figure 6.14: Ilustrasi Membaca nilai Plot dari MFCC

6. One-hot Encoding

penggunaan One-hot Encoding adalah dengan membaca data yang memiliki nilai 1 sebagai nilai tinggi atau bisa disebut juga sebagai nilai positif dan 0 sebagai nilai rendah yang bermakna juga sebagai nilai negatif. ilustrasi dapat dilihat pada gambar 6.15

| X | PUBG | FORTNITE | APEX | BF5 |
|---------|------|----------|------|-----|
| Gamer 1 | 1 | 0 | 0 | 0 |
| Gamer 2 | 0 | 1 | 0 | 0 |
| Gamer 3 | 0 | 0 | 1 | 0 |
| Gamer 4 | 0 | 0 | 0 | 1 |

Figure 6.15: Ilustrasi One-hot Encoding

7. fungsi dari np.unique dan to_categorical dalam Code Program

fungsi dari NP.UNIQUE adalah untuk membuat data elemen menjadi nilai yang bersifat unik dalam artian (Array), ilustrasi dapat dilihat pada gambar 6.16

```
>>> np.unique([1, 1, 2, 2, 3, 3])
array([1, 2, 3])
>>> a = np.array([[1, 1], [2, 3]])
>>> np.unique(a)
array([1, 2, 3])
```

Figure 6.16: Ilustrasi np.unique

sedangkan perintah to_categorical adalah untuk membuat data integer yang terdeteksi untuk diubah menjadi data matrix biner. ilustrasi dapat dilihat pada gambar 6.17

```
# Consider an array of 5 labels out of a set of 3 classes {0, 1, 2}:
> labels
array([0, 2, 1, 2, 0])
# `to_categorical` converts this into a matrix with as many
# columns as there are classes. The number of rows
# stays the same.
> to_categorical(labels)
array([[ 1.,  0.,  0.],
       [ 0.,  0.,  1.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.],
       [ 1.,  0.,  0.]], dtype=float32)
```

Figure 6.17: Ilustrasi to_categorical

8. fungsi dari Sequential

fungsi dari Sequential dari code program adalah untuk membagi data - data agar dapat dianalisis oleh sistem lebih mudah, misalkan dari data 100 dibagi prosesnya menjadi 4 yaitu 25 perproses, seperti yang terdapat pada gambar 6.18 sehingga hasil yang didapatkan akan lebih baik lagi.

| X | B1 | B2 | B3 | B4 |
|----|------|-------|-------|--------|
| N1 | 1>25 | 26>50 | 56>75 | 76>100 |

Figure 6.18: Ilustrasi fungsi Sequential

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 sistem (0,5) c. Lengkap dan bersistem (1) |
| 6 | Pemanfaatan Instrumen Pendukung | Pemanfaatan Instrumen Pendukung seperti gambar dan tabel | 1 | a. Takermanfaatkan (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] Heryn Februariyanti and Eri Zuliarso. Klasifikasi dokumen berita teks bahasa indonesia menggunakan ontologi. *Dinamik*, 17(1), 2012.
- [4] Deny Kurniawan. Regresi linier. *R-Foundation for Statistical Computing*. Vienna, Austria, 17, 2008.
- [5] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.
- [6] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.