# Modul Praktikum
# Kecerdasan Buatan


Eriskiannisa Febrianty Luchia A
1194013

Applied Bachelor of Informatics Engineering
Program Studi D4 Teknik Informatika

Applied Bachelor Program of Informatics Engineering
*Politeknik Pos Indonesia*

Bandung 2022

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

# Acknowledgements

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

# Abstract

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

# Contents

# List of Figures

# Chapter 1

# Mengenal Kecerdasan Buatan dan Scikit-Learn

## 1.1 Kecerdasan Buatan

### 1.1.1 Definisi Kecerdasan Buatan

Artificial Inteligence (AI) atau dalam bahasa Indonesia berarti kecerdasan buatan merupakan sebuah cabang dalam bisnis sains komputer yang mengkaji tentang bagaimana sebuah komputer dapat memiliki kemampuan mirip dengan kemampuan yang dimiliki manusia. Kecerdasan buatan juga dapat didefinisikan sebagai simulasi dari kecerdsan yang dimiliki oleh manusia dan dimodelkan dalam bentuk mesin yang di program agar dapat berpikir layaknya manusia. Dengan kata lain kecerdasan buatan atau AI merupakan sistem yang dapat melakukan suatu pekerjaan yang umumnya memerlukan kecerdasan manusia untuk dapat menyelesaikan pekerjaan tersebut.

Dalam perkembangannya, AI atau kecerdasan buatan sangat membantu manusia dan dapat ditemui di dalam kehidupan sehari – hari seperti Asisten Virtual Google yang digunakan untuk memudahkan user untuk menemukan berbagai hal di dalam perangkat dan hampir semua perangkat komputer menerapkan kecerdasan buatan sehingga semakin canggih kemampuan komputer dalam memperbarui pengetahuannya.

### 1.1.2 Sejarah Kecerdasan Buatan

Artificial intelligence bermula sejak kemunculan komputer sekitar tahun 1940 dan 1950. Pada masa ini perhatian di fokuskan pada kemampuan sebuah komputer untuk dapat mengerjakan sesuatu yang dapat dilakukan manusia. Pada tahun 1943, McMulloh dan Pitts mengusulkan sebuah model matematis dengan nama perceptron dari neuron di dalam otak. Mereka dapat menunjukkan bagaimana sebuah neuron menjadi

aktif dan neuron tersebut mampu belajar memberikan aksi yang berbeda terhadap waktu dari input yang diberikan. Selanjutnya pada tahun 1950, di dalam paper Alan Turing yang berjudul Computing Machineri and Intelligence mendiskusikan syarat sebuah mesin dianggap cerdas dan beranggapan bahwa mesin yang bisa berperilaku seperti manusia dapat dianggap cerdas.

Pada tahun 1956. John McCarthy dari Massacuhetts Institute of Technology (MIT) menyelenggarakan sebuah konferensi Dartmouth yang mengumpulkan ahli komputer dan mempertemukan para pendiri dalam AI dengan nama The Dartmouth summer research project on artificial intelligence dimana John McCarthy mengusulkan definisi AI yaitu AI merupakan cabang dari ilmu komputer yang berfokus pada pengembangan komputer untuk dapat memiliki kemampuan dan berperilaku seperti manusia. Konferensi tersebut dianggap sebagai awal mula kelahiran kecerdasan buatan.

## 1.2 Scikit-Learn

### 1.2.1 Supervised Learning

Supervised Learning merupakan teknik belajar model dimana sebuah model akan diberi data dan setiap data tersebut akan diberi label. Dalam supervised learning terdapat data training yang menjadi tempat semua data dimana data training berupa input data atau target data yang diinginkan kemudian dilatih untuk dapat melakukan prediksi dalam menjawab target data. Data tersebut akan diuji dan dibandingkan dengan prediksi pada data test. Supervised learning juga dapat didefinisikan sebagai operasi machine learning yang digunakan untuk data dimana ada pemetaan yang tepat antara data input dan output. Kumpulan data akan diberi label yang berarti algoritma akan mengidentifikasi secara eksplisit dan akan melakukan prediksi atau klasifikasi yang sesuai.

Contoh Penerapan Supervised Learning adalah aplikasi pengenalan wajah atau Jaringan Syaraf Tiruan (JST). Convolutional Neural Networks (CNN) adalah jenis JST yang digunakan untuk mengidentifikasi wajah. Jika ada kesamaan antara input atau masukkan nya di dalam database maka akan dihasilkan sebuah output.

Algoritma supervised learning digunakan untuk menyelesaikan persoalan yang terkait dengan:

1. Classification (klasifikasi)
   Classification (klasifikasi) merupakan teknik yang digunakan untuk mengklasi-

fikasikan data yang belum berlabel atau mengelompokkan data berdasarkan ciri persamaan dan perbedaannya. Algoritma klasifikasi juga digunakan untuk memprediksi fitur kategori.

2. Regression (regresi)
   Regression (regresi) merupakan teknik untuk mengidentifikasi sebuah relasi serta hubungan diantara dua variable atau lebih. Regresi digunakan untuk memprediksi nilai atau fitir kontinu. Pada regresi akan dilihat hubungan sebab dan akibat. Variable sebab biasanya dikenal dengan variabel independent (X) dan variable akibat biasanya dikenal dengan variabel dependent (Y).

### 1.2.2 Unsupervised Learning

Unsupervised Learning merupakan operasi machine learning yang mempelajari sebuah pola data tanpa memerlukan target data. Unsupervised Learning hanya memerlukan data input dan menemukan pola serta insight penting dari data. Proses ini biasa disebut dengan data mining. Pada Unsupervised Learning data tidak memiliki label secara eksplisit dan mampu belajar dari data dengan menemukan pola implisit. Unsupervised Learning tidak menggunakan data training dan hanya bergantung pada data test sehingga tidak dapat dilakukan evaluasi pada model. Unsupervised Learning dievaluasi secara subjektif untuk dapat mengetahui bahwa prediksi yang dilakukan sudah sesuai.

Unsupervised Learning merupakan salah satu tipe dari algoritma machine learning yang digunakan untuk menarik kesimpulan dari sebuah data set dan mempelajari data berdasarkan kedekatannya atau disebut dengan clustering. Clustering merupakan sebuah metode yang digunakan untuk pengelompokan secara otomatis dengan membagi kumpulan data menjadi beberapa kelompok sesuai kesamaannya.

### 1.2.3 Data Set

Dataset merupakan objek yang merepresentasikan data dan relasi yang ada di dalam memory. Struktur dari Dataset hampir sama dengan database, namun dataset berisi koleksi data table dan data relation. Dataset akan tetap berada di dalam memori dan data yang ada di dalamnya dapat dimanipulasi dan di update tanpa bergantung pada database asalnya.

### 1.2.4 Training Set

Training set merupakan set yang digunakan oleh algoritma klasifikasi. Training set juga merupakan bagian dari dataset yang digunakan untuk membuat prediksi dan menjalankan fungsi dari algoritma machine learning. Training set digunakan untuk algoritma klasifikasi seperti decision tree, Bayesian, neural network dan svm untuk dapat membentuk sebuah model classifier. Model ini akan merepresentasikan pengetahuan yang digunakan untuk prediksi kelas data baru yang belum ada.

### 1.2.5 Testing Set

Testing set merupakan bagian dari dataset yang akan di tes untuk dapat melihat ke akuratan atau performanya dengan kata lain, testing set digunakan untuk menguji peforma serta kebenaran dari model yang bersangkutan. Testing set akan mengukur sejauh mana classifier berhasil melakukan klasifikasi dengan benar.

# Chapter 2

# Membangun Model Prediksi

## 2.1 Teori

### 2.1.1 Binary Classification

Binary Classification merupakan klasifikasi yang memiliki 2 label kelas. Biasanya melibatkan satu kelas yang merupakan keadaan normal dan kelas lainnya yang merupakan keadaan abnormal atau dapat berarti binary classification berupa kelas positif dan kelas negatif. Contohnya, pada email terdeteksi spam email, ada keadaan dimana email tersebut dapat berupa spam atau bukan. Misalnya bukan spam berarti keadaan normal dan spam berarti keadaan abnormal. Kelas dengan keadaan normal atau positif diberi label kelas 0 dan kelas dengan keadaan abnormal atau negatif diberi label kelas 1.
Algoritma yang digunakan untuk binary classification yaitu :

1. Logistic Regression
2. K-Nearest Neighbors
3. Decision Trees
4. Support Vector Machine
5. Naive Bayes

### 2.1.2 Supervised Learning, Unsupervised Learning dan Clustering

1. Supervised Learning
   Supervised Learning merupakan teknik belajar model dimana sebuah model

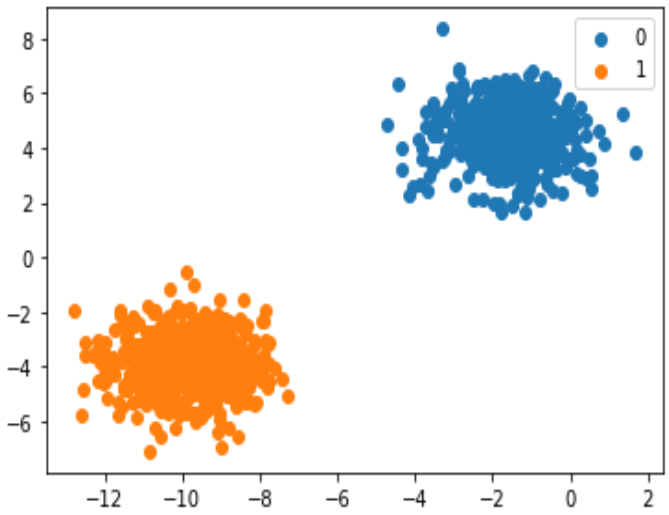


Figure 2.1: Binary Classification

akan diberi data dan setiap data tersebut akan diberi label. Dalam supervised learning terdapat data training yang menjadi tempat semua data dimana data training berupa input data atau target data yang diinginkan kemudian dilatih untuk dapat melakukan prediksi dalam menjawab target data. Data tersebut akan diuji dan dibandingkan dengan prediksi pada data test. Supervised learning juga dapat didefinisikan sebagai operasi machine learning yang digunakan untuk data dimana ada pemetaan yang tepat antara data input dan output. Kumpulan data akan diberi label yang berarti algoritma akan mengidentifikasi secara eksplisit dan akan melakukan prediksi atau klasifikasi yang sesuai.

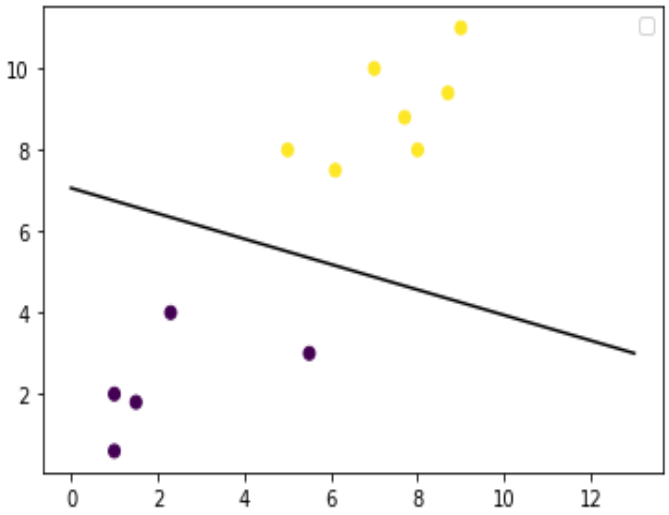


Figure 2.2: Supervised Learning : SVM Model Using Linear Kernel

2. Unsupervised Learning

   Unsupervised Learning merupakan operasi machine learning yang mempelajari sebuah pola data tanpa memerlukan target data. Unsupervised Learning hanya memerlukan data input dan menemukan pola serta insight penting dari data. Proses ini biasa disebut dengan data mining. Pada Unsupervised Learning data tidak memiliki label secara eksplisit dan mampu belajar dari data dengan menemukan pola implisit. Unsupervised Learning tidak menggunakan data training dan hanya bergantung pada data test sehingga tidak dapat dilakukan evaluasi pada model. Unsupervised Learning dievaluasi secara subjektif untuk dapat mengetahui bahwa prediksi yang dilakukan sudah sesuai.

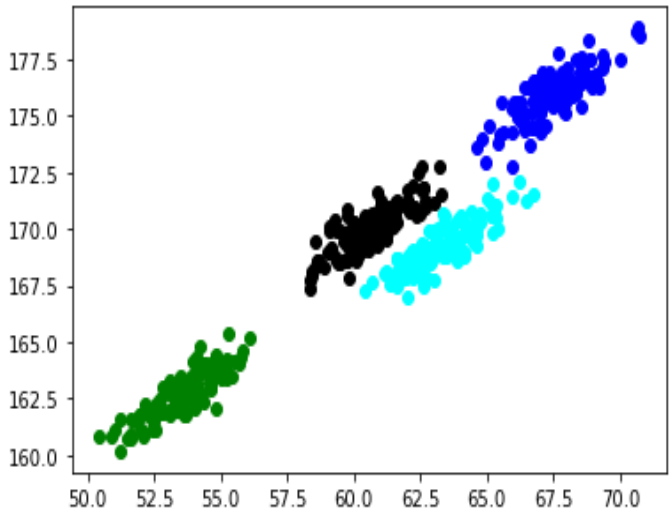


Figure 2.3: Unsupervised Learning : Gaussian Mixture Model

3. Clustering

   Clustering merupakan sebuah teknik machine learning yang digunakan untuk mengelompokkan data tidak berlabel atau merupakan sebuah metode yang digunakan untuk pengelompokkan data secara otomatis dengan membagi kumpulan data menjadi beberapa kelompok sesuai kesamaannya. Algoritma pengelompokkan digunakan untuk memproses objek data yang tidak di klasifikasi ke dalam kelompok yang diwakili dengan struktur atau sebuah pola dalam informasi

### 2.1.3 Evaluasi dan Akurasi

Evaluasi merupakan kegiatan yang dilakukan untuk mengukur seberapa baik sebuah model dapat bekerja dengan menghitung akurasinya. Akurasi merupakan ukuran

atau persentase data yang diklasifikasikan dengan benar. Akurasi klasifikasi juga dapat berarti membagi jumlah prediksi benar terhadap total prediksi. Dalam model klasifikasi, dapat diprediksi nilai terbesar dan memberikan akurasi yang tinggi serta model yang dihasilkan dapat memprediksi nilai yang salah. Sehingga dalam hal ini dibutuhkan adanya metrik evaluasi yang dapat mengukur peforma dari model klasifikasi yang sudah dibuat. Metrik yang digunakan adalah Precision, Recall dan Confusion Matrix.

### 2.1.4 Confusion Matrix

Confusion Matrix atau disebut juga dengan Error Matrix pada dasarnya akan memberikan informasi mengenai perbandingan hasil dari klasifikasi yang dilakukan oleh model atau sistem dengan hasil sebenarnya. Confusion Matrix berbentuk tabel matrix yang akan menggambarkan kinerja dari model klasifikasi pada data uji yang nilai sebenarnya diketahui.

**Berikut adalah contoh dari confusion matrix :**

1. Accuracy
2. Precision (Positive Predictive Value)
3. Recall atau Sensitivity (True Positive Rate)

**Cara membuat dan membaca confusion matrix**

1. Tentukan pokok permasalahan atau menggunakan dataset, misalnya data pasien covid
2. Membagi dataset serta membuat model prediksi menggunakan Decision Tree. Dataset dibagi menjadi dua yaitu data latiih dan data uji. Data latih digunakan untuk melatih model yang dibuat dan evaluasi akan dilakukan pada data uji.
3. Evaluasi model menggunakan confusion matrix yaitu untuk mengetahui keakuratan model yang sudah dibuat menggunakan performance metrics seperti: accuracy, recall, dan precision.

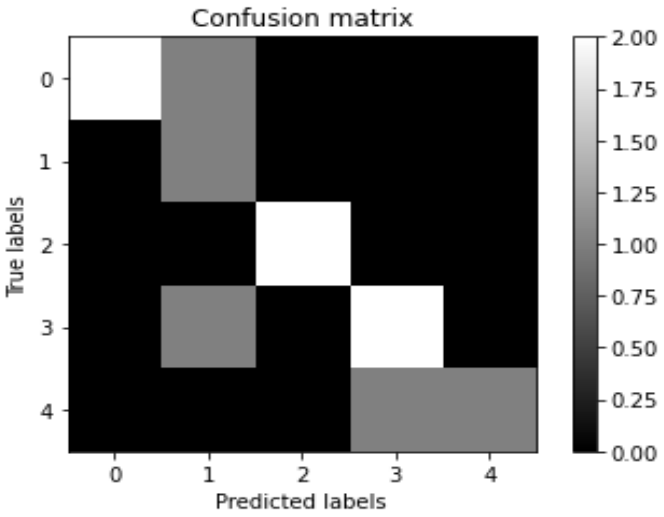


Figure 2.4: Confusion Matrix

### 2.1.5 K-fold cross validation

K-fold cross validation merupakan model evaluasi atau prosedur yang akan memisahkan antara data training dan data testing. K-fold cross validation dapat di definisikan sebagai pengujian cross validation yang digunakan untuk menilai kinerja dari sebuah metode algoritma dengan membagi sampel data secara acak dan mengelompokkan data tersebut sebanyak nilai K k-fold.

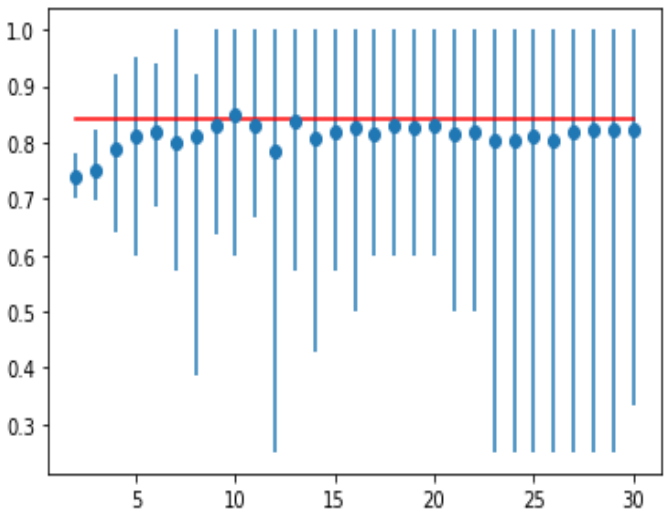


Figure 2.5: K-fold cross validation

**Cara kerja K-fold cross validation :**

1. Tentukan instance, dibagi menjadi N bagian atau misalnya ada 10 data dan akan dilakukan K-fold cross validation pada data tersebut.

2. Data dibagi menjadi data testing untuk pengujian pada model dan data training untuk melatih model.

3. Menentukan nilai K, misalnya dalam hal ini ditentukan nilai K = 10 dimana data tersebut nantinya akan ada 10 lipatan atau disebut dengan fold

### 2.1.6 Decision Tree

Decision Tree merupakan metode penbelajaran non parametrik yang digunakan untuk klasifikasi dan regresi. Tujuan dari decision tree adalah membuat model yang akan memprediksi nilai variable dengan mempelajari aturan keputusan sederhana yang disimpulkan dari sebuah fitur data. Decision tree merupakan struktur yang sama seperti diagram alur dimana simpul internal sebagai fitur atau atribut, cabang sebagai aturan dan keputusan, serta setiap simpul daun akan mewakili hasilnya.

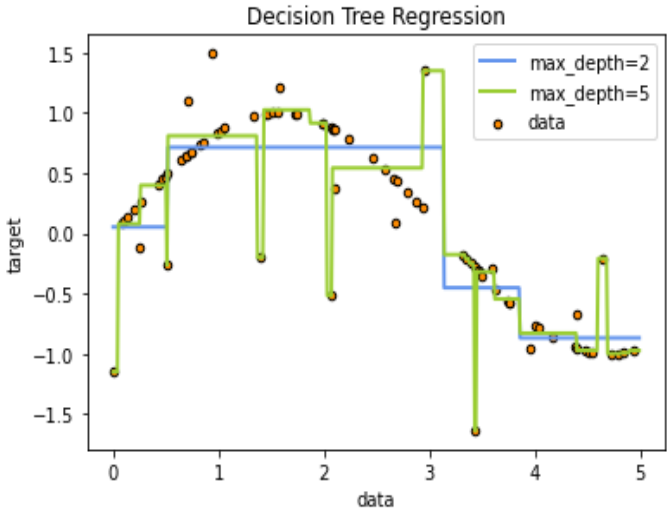


Figure 2.6: Decision Tree

### 2.1.7 Information Gain dan Entropi

1. Entropi
   Sebuah objek yang di akan di klasifikasikan ke dalam decision tree harus di uji nilai entropinya. Entropi merupakan ukuran dari informasi yang dapat mengetahui karakteristik dari dari impuryt ,dan homogenity dari sekumpulan data.

Entropi juga merupakan jumlah bit yang diperkirakan akan dibutuhkan untuk mengekstrak suatu kelas dari data acak pada suatu ruang sampel. Dari nilai entropi tersebut kemudian akan dihitung information gain masing-masing atribut.

2. Information Gain
   Kemudian setelah mendapatkan nilai entropi untuk suatu kumpulan data, maka akan diukur efektivitas nya atau disebut dengan information gain. Information gain digunakan untuk mengukur seberapa besar relevan atau pengaruh sebuat feature terhadap hasil pengukuran. Information Gain dikenal juga dengan sebutan Mutual Information dalam kasus untuk mengetahui dependency antara dua variable (x,y).

## 2.2 Praktikum

### 2.2.1 Scikit-Learn

1. Load Dataset student-mat.csv
   Digunakan untuk import module pandas dan mendefinisikan variable "bandung" yang akan memanggil dataset yang diambil dari student-mat.csv.

```
In [17]: import pandas as pd
    ...: bandung = pd.read_csv('D:\File_Kuliah\student-mat.csv', sep=';')
    ...: len(bandung)
    ...: print(len(bandung))
395
```

Figure 2.7: Load Dataset

2. Generate Binary Label
   Bagian ini akan mendeklarasikan pass/fail data berdasarkan G1+G2+G3 dengan ketentuan nilai pass = 30 dan pada variable bandung dideklarasikan jika baris dengan G1+G2+G3 ditambahkan, dan hasilnya sama dengan 35 maka axis nya 1.

3. Use one-hot encoding on categorical columns
   One-hot encoding merupakan proses dimana sebuah variable kategorikal dikonversikan menjadi bentuk yang dapat disediakan oleh algoritma Machine Learning untuk dapat melakukan pekerjaan yang lebih baik dalam memprediksi. Disini menggunakan fungsi panda pdgetdummies untuk jenis kelamin, sekolah,

```
In [18]: bandung = pd.read_csv('D:\File_Kuliah\student-mat.csv', sep=';')
    ...: bandung['pass'] = bandung.apply(lambda row: 1 if (row['G1']+row['G2']+row['G3'])
    ...:                     >= 35 else 0, axis=1)
    ...: bandung = bandung.drop(['G1', 'G2', 'G3'], axis=1)
    ...: bandung.head()
    ...: print(bandung.head())
  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 2.8: Generate Binary Label

alamat dan lainnya. Metode head ini digunakan untuk mengembalikan baris n atas 5 secara default dari frame atau seri datanya.

```
In [19]: bandung = pd.get_dummies(bandung, columns=['sex', 'school', 'address','famsize',
'Pstatus', 'Mjob',
    ...:                                 'Fjob','reason', 'guardian', 'schoolsup',
'famsup',
    ...:                                 'paid', 'activities', 'nursery', 'higher',
'internet',
    ...:                                 'romantic'])
    ...: bandung.head()
    ...: print(bandung.head())
   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 2.9: Use one-hot encoding on categorical columns

4. Shuffle Rows

   Digunakan untuk mengembalikkan sample secara acak item dari objek. Terdapat train dan test yang digunakan untuk membagi train, test dan kemudian dibagi lagi train ke validasi dan test. kemudian di import sebuah modul numpy sebagai np yang digunakan untuk mengembalikan nilai passing dari pelajar dan dari keseluruhan dataset dengan menggunakan print.

5. Fit a decision tree

   Import modul tree dari library scikit-learn. Kemudian definisikan variable bandung menggunakan decision classifier. Di dalam variable bandung terdapat criterion yang merupakan suatu fungsi untuk mengukur kualitas split. Agar decision tree classifier dapat di jalankan maka gunakan perintah fit

6. Visualize tree

   Graphviz merupakan perangkat lunak visualisasi grafik objek open source. Vi-

```
In [21]: bandung = bandung.sample(frac=1)
    ...: # split training and testing data
    ...: bandung_train = bandung[:500]
    ...: bandung_test = bandung[500:]
    ...:
    ...: bandung_train_att = bandung_train.drop(['pass'], axis=1)
    ...: bandung_train_pass = bandung_train['pass']
    ...:
    ...: bandung_test_att = bandung_test.drop(['pass'], axis=1)
    ...: bandung_test_pass = bandung_test['pass']
    ...:
    ...: bandung_att = bandung.drop(['pass'], axis=1)
    ...: bandung_pass = bandung['pass']
    ...: # number of passing students in whole dataset:
    ...: import numpy as np
    ...: print("Passing: %d out of %d (%.2f%%)" % (np.sum(bandung_pass),
len(bandung_pass),
    ...:                                           100*float(np.sum(bandung_pass)) /
len(bandung_pass)))
Passing: 166 out of 395 (42.03%)
```

Figure 2.10: Shuffle Rows

```
In [22]: from sklearn import tree
    ...: solo = tree.DecisionTreeClassifier(criterion="entropy", max_depth=5)
    ...: solo = solo.fit(bandung_train_att, bandung_train_pass)
```

Figure 2.11: Fit a decision tree

sualisasi grafik merupakan cara untuk mewakili informasi struktural sebagai diagram grafik dan jaringan abstrak. treeexportgraphviz merupakan sebuah fungsi yang akan menghasilkan representasi Graphviz dari decision tree,kemudian ditulis kedalam out file, sehingga akan ditampilkan sebuah diagram grafik bercabang.

```
In [23]: import graphviz
    ...: dot_data = tree.export_graphviz(solo, out_file=None, label="all",
    ...:                                 impurity=False, proportion=True,
    ...:                                 feature_names=list(bandung_train_att),
    ...: class_names=["fail", "pass"], filled=True, rounded=True)
    ...: graph = graphviz.Source(dot_data)
    ...: graph
Out[23]:
```

Figure 2.12: Visualize tree

7. Save Tree

   TREEEXPORTGRAPHVIZ merupakan sebuah fungsi yang akan menghasilkan representasi graphviz dari decision tree yang kemudian akan di tulis ke dalam sebuah outfile. Di dalam file tersebut akan disimpan classifier nya kemudian mengekspor file tersebut dengan namastudent performance kedalam folder tujuan kemudian jika salah maka akan mengembalikan nilai fail.

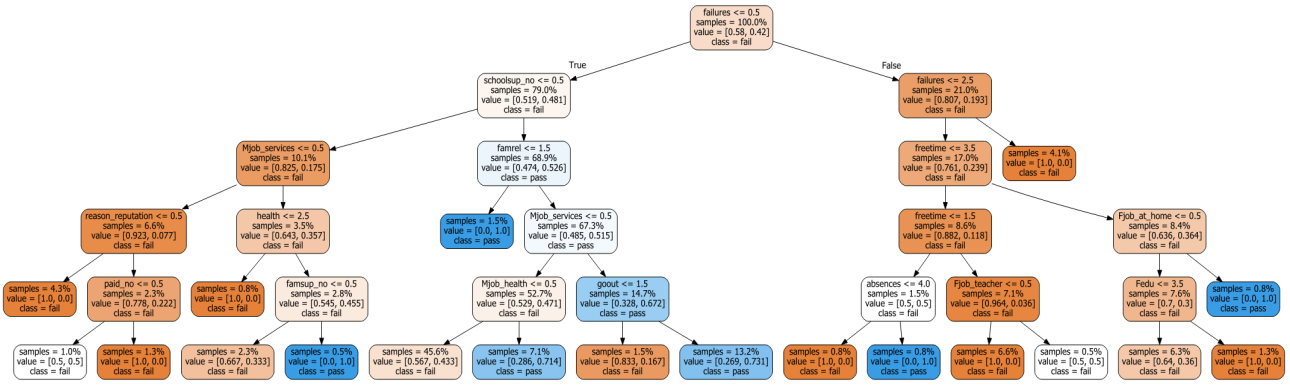


Figure 2.13: Visualize tree - graph

```
In [24]: tree.export_graphviz(solo, out_file="student-performance.dot",
    ...:                     label="all", impurity=False,
    ...:                     proportion=True,
    ...:                     feature_names=list(bandung_train_att),
    ...:                     class_names=["fail", "pass"],
    ...:                     filled=True, rounded=True)
```

Figure 2.14: Save Tree

8. Score

   Score disebut juga sebagai prediksi dan merupakan proses yang akan menghasilkan nilai berdasarkan pada model pembelajaran mesin yang terlatih dan diberi beberapa data input baru. Nilai dibuat untuk mewakili prediksi nilai di masa depan atau juga dapat mewakili kategori serta hasil yang mungkin. Dalam hal ini variable solo akan memprediksi nilai bandung test att dan test pass.

9. Evaluasi Score - Cross Val Score

   Pada Evaluasi Score, di script ini akan mengevaluasi score dengan menggunakan validasi silang. Dimana variable jakarta disini berisi cross val score yang merupakan fungsi pembantu pada estimator serta dataset. Kemudian akan ditampilkan score rata-rata dan kurang lebih dua standar deviasi yang mencakup 95 persen dari score.

```
In [10]: from sklearn.model_selection import cross_val_score
    ...: jakarta = cross_val_score(solo, bandung_att, bandung_pass, cv=5)
    ...: # show average score and +/- two standard deviations away
    ...: #(covering 95% of scores)
    ...: print("Accuracy: %0.2f (+/- %0.2f)" % (jakarta.mean(), jakarta.std() * 2))
Accuracy: 0.58 (+/- 0.08)
```

Figure 2.15: Evaluasi Score - Cross Val Score

10. Max Depth

    Script berikut akan menunjukkan bahwa semakin banyak tree maka semakin banyak perpecahan yang dimiliki dan akan lebih banyak menangkap informasi dari data. Variable solo disini akan mendefinisikan tree kemudian variable jakarta akan mengevaluasi score nya dengan validasi silang. Kemudian akan di definisikan decision tree dengan kedalaman mulai dari 1 hingga 20 dan merencanakan pelatihan dan menguji skor auc.

```
In [11]: for max_depth in range(1, 20):
    ...:     solo = tree.DecisionTreeClassifier(criterion="entropy",
    ...:                                        max_depth=max_depth)
    ...:     jakarta = cross_val_score(solo, bandung_att, bandung_pass, cv=5)
    ...:     print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" %
    ...:           (max_depth, jakarta.mean(), jakarta.std() * 2) )
Max depth: 1, Accuracy: 0.58 (+/- 0.01)
Max depth: 2, Accuracy: 0.58 (+/- 0.07)
Max depth: 3, Accuracy: 0.56 (+/- 0.07)
Max depth: 4, Accuracy: 0.56 (+/- 0.05)
Max depth: 5, Accuracy: 0.58 (+/- 0.08)
Max depth: 6, Accuracy: 0.58 (+/- 0.07)
Max depth: 7, Accuracy: 0.57 (+/- 0.02)
Max depth: 8, Accuracy: 0.55 (+/- 0.09)
Max depth: 9, Accuracy: 0.57 (+/- 0.06)
Max depth: 10, Accuracy: 0.57 (+/- 0.04)
Max depth: 11, Accuracy: 0.57 (+/- 0.07)
Max depth: 12, Accuracy: 0.58 (+/- 0.07)
Max depth: 13, Accuracy: 0.59 (+/- 0.08)
Max depth: 14, Accuracy: 0.58 (+/- 0.01)
Max depth: 15, Accuracy: 0.60 (+/- 0.03)
Max depth: 16, Accuracy: 0.59 (+/- 0.05)
Max depth: 17, Accuracy: 0.59 (+/- 0.05)
Max depth: 18, Accuracy: 0.58 (+/- 0.02)
Max depth: 19, Accuracy: 0.58 (+/- 0.04)
```

Figure 2.16: Max Depth

11. Depth In Range

    Depth acc membuat array kosong dengan mengembalikan array baru menggunakan bentuk dan tipe yang diberikan, tanpa menginisialisasi entri. Dengan 19 sebagai bentuk array kosong, 3 sebagai output data-type dan float urutan kolom utama (gaya Fortran) dalam memori. Variabel solo yang akan melakukan split score dan jakarta akan mengvalidasi score secara silang. Kemudian jakarta std yaitu menghitung standar deviasi dari data yang diberikan (elemen array) di sepanjang sumbu yang ditentukan (jika ada).

12. Matplotlib

    Pada Script berikut, akan di import sebuah library dari matplotlib yaitu pylot sebagai plt, fig dan ax yang menggunakan subplots untuk dapat membuat gambar serta satu set subplot. axerrorbar dalam script akan membuat error bar kemudian membuat sebuah grafik yang akan ditampilkan menggunakan perintah show.

```
In [12]: depth_acc = np.empty((19,3), float)
    ...: i = 0
    ...:
    ...: for max_depth in range(1, 20):
    ...:     solo = tree.DecisionTreeClassifier(criterion="entropy",
    ...:                                        max_depth=max_depth)
    ...:     jakarta = cross_val_score(solo, bandung_att, bandung_pass, cv=5)
    ...:     depth_acc[i,0] = max_depth
    ...:     depth_acc[i,1] = jakarta.mean()
    ...:     depth_acc[i,2] = jakarta.std() * 2
    ...:     i += 1
    ...: depth_acc
Out[12]:
array([[1.00000000e+00, 5.79746835e-01, 1.01265823e-02],
       [2.00000000e+00, 5.82278481e-01, 6.60172395e-02],
       [3.00000000e+00, 5.62025316e-01, 6.71721477e-02],
       [4.00000000e+00, 5.69620253e-01, 6.40461298e-02],
       [5.00000000e+00, 5.74683544e-01, 8.10126582e-02],
       [6.00000000e+00, 5.82278481e-01, 7.67886121e-02],
       [7.00000000e+00, 5.79746835e-01, 3.35860738e-02],
       [8.00000000e+00, 5.51898734e-01, 6.52359429e-02],
       [9.00000000e+00, 5.67088608e-01, 9.11392405e-02],
       [1.00000000e+01, 5.62025316e-01, 5.68353021e-02],
       [1.10000000e+01, 5.84810127e-01, 6.28337906e-02],
       [1.20000000e+01, 5.74683544e-01, 4.41407488e-02],
       [1.30000000e+01, 5.84810127e-01, 6.28337906e-02],
       [1.40000000e+01, 5.94936709e-01, 5.77304013e-02],
       [1.50000000e+01, 5.74683544e-01, 4.41407488e-02],
       [1.60000000e+01, 6.05063291e-01, 6.86818226e-02],
       [1.70000000e+01, 6.00000000e-01, 5.21297729e-02],
       [1.80000000e+01, 5.82278481e-01, 3.20230649e-02],
       [1.90000000e+01, 5.84810127e-01, 5.16356406e-02]])
```

Figure 2.17: Depth In Range

```
In [31]: import matplotlib.pyplot as plt
    ...: fig, ax = plt.subplots()
    ...: ax.errorbar(depth_acc[:,0], depth_acc[:,1], yerr=depth_acc[:,2])
    ...: plt.show()
```

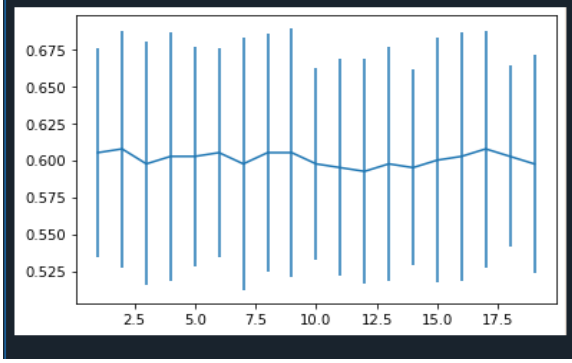


Figure 2.18: Matplotlib

## 2.3 Penanganan Error

### 2.3.1 Error Graphviz

1. Berikut merupakan error yang didapat setelah menjalankan graphviz pada spyder

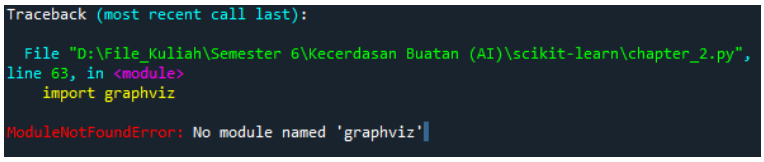


Figure 2.19: Error Graphviz

**Berikut merupakan Solusinya**

1. Install conda install -c anaconda pydot

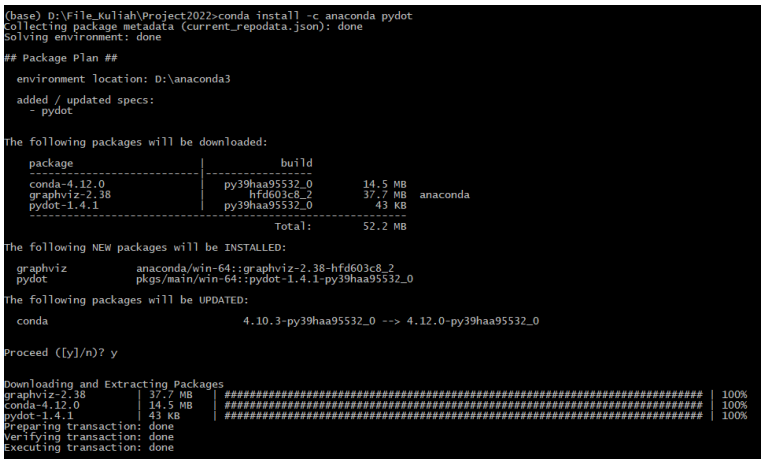


Figure 2.20: Solusi 1

2. install conda install -c anaconda python-graphviz

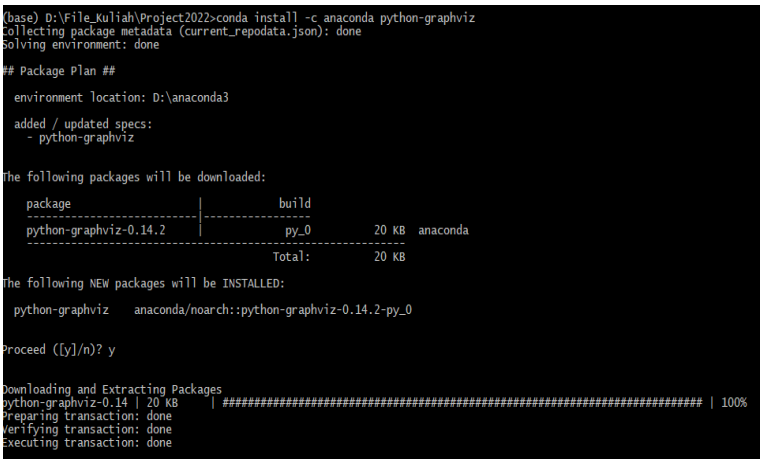


Figure 2.21: Solusi 2

# Chapter 3

# Prediksi dengan Random Forest

## 3.1 Teori

### 3.1.1 Random Forest

Random forest terdiri dari kumpulan decision tree. Random forest merupakan algoritma supervised learning yang digunakan untuk klasifikasi dan regresi. Random forest dapat menangani kumpulan data yang berisi variable kontinu seperti dalam kasus regresi dan variable kategoris dalam kasus klasifikasi. Random forest bekerja sepeti enseble dimana enseble berarti menggabungkan beberapa model menjadi kumpulan model yang digunakan untuk membuat prediksi. Random forest akan memilih pengamatan secara acak kemudian membangun sebuah decision tree dan mengambil hasil rata-ratanya.

### 3.1.2 Dataset

Dataset merupakan sekumpulan data dimana data terabut berasal dari informasi di masa lalu yang dikelola menjadi sebuah informasi untuk dapat melakukan data mining. Dataset berisi lebih dari satu variable yang digunakan untuk klasifikasi.
**Cara Membaca Dataset dan Arti Setiap File dan isi Field Masing-masing File :**

1. Menggunakan library Pandas pada python untuk membaca dataset dengan format text file.

2. Kemudian buat variable baru misalnya "dataset" yang berisi perintah untuk membaca file dataset yang digunakan

```
In [17]: import pandas as pd
    ...: bandung = pd.read_csv('D:\File_Kuliah\student-mat.csv', sep=';')
    ...: len(bandung)
    ...: print(len(bandung))
395
```

Figure 3.1: Load Dataset

### 3.1.3 Cross Validation

Cross validation merupakan metode untuk mengevaluasi dan membandingkan algoritma dengan membagi data menjadi dua yaiti data latih dan data uji. Cross validation (CV) adalah salah satu teknik yang digunakan untuk menguji keefektifan suatu model dan merupakan prosedur pengambilan sampel ulang yang digunakan untuk mengevaluasi suatu model jika memiliki data yang terbatas. Untuk dapat melakukan CV maka perlu menyisihkan sampel/sebagian data yang tidak digunakan untuk melatih model, kemudian menggunakan sampel ini untuk pengujian/validasi. Bentuk dasar dari cross-validation adalah k-fold cross-validation.

### 3.1.4 Arti Score 44% Pada Random Forest, 27% Pada Decission Tree Dan 29% Dari SVM

merupakan presentase keakurasian prediksi yang dilakukan pada saat testing menggunakan label pada dataset yang digunakan. Score akan mendefinisikan aturan evaluasi model, kemudian saat dijalankan akan muncul sebuah persentase yang menunjukan keakurasian atau keberhasilan dari prediksi yang dilakukan. Jika menggunakan Random Forest maka hasilnya 40% dan jika menggunakan Decission Tree hasil prediksinya yaitu 27% dan pada SVM 29%.

### 3.1.5 Confusion Matrix

Confusion Matrix atau disebut juga dengan Error Matrix pada dasarnya akan memberikan informasi mengenai perbandingan hasil dari klasifikasi yang dilakukan oleh model atau sistem dengan hasil sebenarnya. Confusion Matrix berbentuk tabel matrix yang akan menggambarkan kinerja dari model klasifikasi pada data uji yang nilai sebenarnya diketahui.

**Berikut adalah contoh dari confusion matrix :**

1. Accuracy
2. Precision (Positive Predictive Value)

3. Recall atau Sensitivity (True Positive Rate)

**Cara membuat dan membaca confusion matrix**

1. Tentukan pokok permasalahan atau menggunakan dataset, misalnya data pasien covid

2. Membagi dataset serta membuat model prediksi menggunakan Decision Tree. Dataset dibagi menjadi dua yaitu data latiih dan data uji. Data latih digunakan untuk melatih model yang dibuat dan evaluasi akan dilakukan pada data uji.

3. Evaluasi model menggunakan confusion matrix yaitu untuk mengetahui keakuratan model yang sudah dibuat menggunakan performance metrics seperti: accuracy, recall, dan precision.

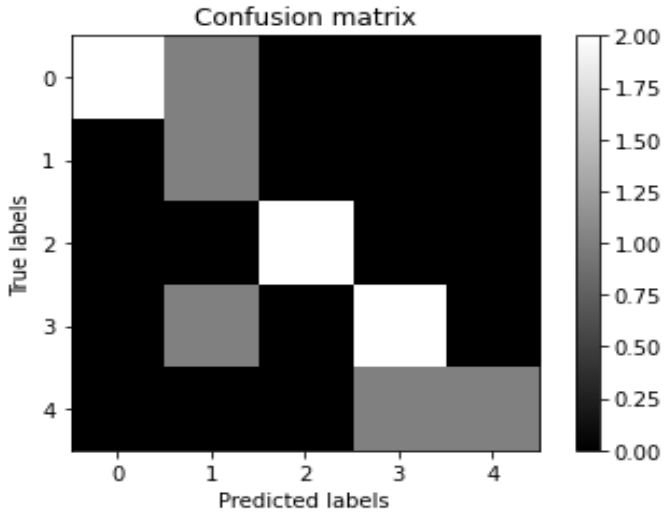


Figure 3.2: Confusion Matrix

### 3.1.6 Voting Pada Random Forest

Voting merupakan suara untuk setiap target yang diprediksi pada saat melakukan Random Forest. Target prediksi dengan voting tertinggi digunakan sebagai prediksi akhir dari algoritma random forest.

## 3.2 Praktikum

### 3.2.1 Aplikasi Sederhana Menggunakan Pandas

**Penjelasan Code Aplikasi Sederhana pandas perbaris :**

```
In [3]: import pandas as pd
   ...: data = {'Nama': ['Agus', 'Budi', 'Ciko', 'Danang', 'Eko'],
   ...:         'NPM': ['101901', '101902', '101903', '101904', '101904'],
   ...:         'Angkatan': ['2019', '2019', '2019', '2019', '2019']}
   ...: frame = pd.DataFrame(data)
   ...: frame
Out[3]:
     Nama     NPM Angkatan
0    Agus  101901     2019
1    Budi  101902     2019
2    Ciko  101903     2019
3  Danang  101904     2019
4     Eko  101904     2019
```

Figure 3.3: Aplikasi sederhana menggunakan pandas

- Baris Pertama Yaitu import library pandas kemudian di inisialisasi menjadi pd
- Variable data di definisikan data data untuk kolom nama, kolom npm dan kolom angkatan
- Variable frame akan mengubah data pada variable data disejajarkan dengan baris dan kolom menggunakan pd dataframe
- Peritah frame kemudian dijalankan untuk dapat menampilkan hasil dari dataframe

### 3.2.2 Aplikasi Sederhana Menggunakan Matplotlib

```
In [10]: import matplotlib.pyplot as plt
    ...:
    ...: prodi = ['Matematika', 'Statistika', 'Biologi', 'Kimia', 'Teknik Informatika',
    ...:          'Fisika']
    ...: jumlah_mhs = [108, 134, 167, 155, 120, 97]
    ...:
    ...: plt.figure(figsize=(8,4))
    ...: plt.bar(prodi, jumlah_mhs, color='lightcoral')
    ...:
    ...: plt.title('Jumlah Mahasiswa Per Program Studi', size=16)
    ...: plt.ylabel('Jumlah Mahasiswa', size=14)
    ...: plt.xticks(size=9)
    ...: plt.yticks(size=9)
    ...:
    ...: plt.show()
```

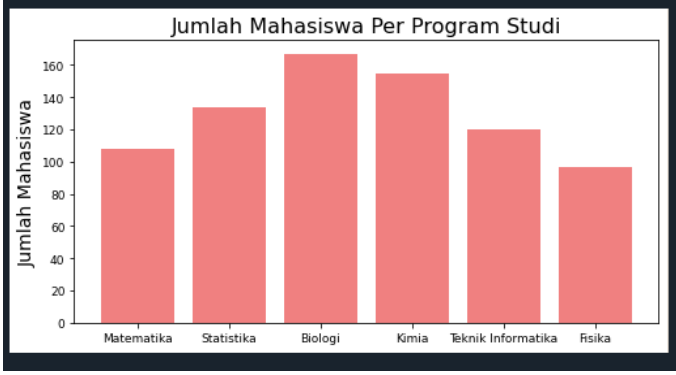


Figure 3.4: Aplikasi sederhana menggunakan matplotlib

**Penjelasan Code Aplikasi Sederhana matplotlib perbaris :**

- Baris Pertama Yaitu import library matplotlib kemudian di inisialisasi menjadi plt
- Variable prodi sebagai sumbu x untuk nama nama prodi
- Variable jumlah mhs sebagai sumbu y untuk jumlah atau banyaknya mahasiswa
- plt figure digunakan untuk mengatur size dan plt bar merupakan fungsi yang digunakan untuk memvisualisasikan bar
- plt title digunakan untuk memberikan judul dan plt ylabel untuk memberikan judul pada sumbu y
- plt yticks dan xticks digunakan untuk mengatur size tulisan pada sumbu y dan x

### 3.2.3 Aplikasi Sederhana Menggunakan Numpy

```
In [15]:
    ...: import numpy as np
    ...: a = np.array ([1, 2, 3, 4])
    ...: print(a)
    ...: b = np.array ([1, 3, 5, 7])
    ...: print(b)
    ...: c = np.array ([2, 4, 6, 8])
    ...: print(c)
    ...:
    ...: #operasi array
    ...: a * b
    ...: print(a*b)
    ...: b + c
    ...: print(b+c)
[1 2 3 4]
[1 3 5 7]
[2 4 6 8]
[ 1  6 15 28]
[ 3  7 11 15]
```

Figure 3.5: Aplikasi sederhana menggunakan numpy

**Penjelasan Code Aplikasi Sederhana numpy perbaris :**

- Baris Pertama Yaitu import library numpy kemudian di inisialisasi menjadi np
- Variable a digunakan sebagi fungsi array yang pertama
- Variable b digunakan sebagai fungsi array yang kedua
- Variable c digunakan sebagai fungsi array yang ketiga

- Perintah print digunakan untuk menampilkan hasil array
- Operasi pada array menggunakan perkalian pada a dan b dan penjumlahan pada b dan c

### 3.2.4 Program Klasifikasi Random Forest

Berikut merupakan output dari percobaan Random Forest yang telah dilakukan :

1. kode berikut digunakan untuk membaca data file txt atau dataset yang akan digunakan. Kemudian output yang dikeluarkan adalah mengembalikan baris n teratas dari data frame imgatt

```
In [18]:
    ...: import pandas as pd
    ...:
    ...: # Membaca data file txt
    ...: # some lines have too many fields (?) , so skip bad lines
    ...: imgatt = pd.read_csv("F:\File_Kuliah\data\image_attribute_labels.txt",
    ...:                      sep='\s+' , header=None ,
    ...:                      error_bad_lines=False ,
    ...:                      warn_bad_lines=False,
    ...:                      usecols = [0 ,1 ,2],
    ...:                      names = ['imgid', 'attid', 'present'])
    ...: # Melihat sebagian data awal
    ...: imgatt.head ()
Out[18]:
   imgid  attid  present
0      1      1        0
1      1      2        0
2      1      3        0
3      1      4        0
4      1      5        1
```

Figure 3.6: Membaca dataset file txt

2. Kode berikut akan menampilkan output banyaknya jumlah baris dan kolom data frame imgatt

```
In [20]:
    ...: imgatt.shape
Out[20]: (3677856, 3)
```

Figure 3.7: Mengetahui jumlah data

3. imgatt2 menggunakan function pivot untuk merubah kolom menjadi baris dan baris menjadi kolom dari data frame sebelumnya.
4. Pada kode berikut digunakan dataset label kemudian akan diberi label pada burung dan menentukan burung tersebut ke dalam spesies apa. di dalam data

```
In [21]:
    ...: imgatt2 = imgatt.pivot (index = 'imgid' , columns = 'attid', values = 'present')
    ...: imgatt2.head ()
    ...: imgatt2.shape
Out[21]: (11788, 312)
```

Figure 3.8: Pivot dataset

yang sebelumnya ada 312 kolom dimana 312 data tersebut memiliki kelompoknya masing-masing. Dalam hal ini akan dimunculkan dua kolom pada variable explorer yaitu imgd dan label yang terdiri dari 11788 baris dan 1 kolom

```
In [26]:
    ...: imglabels = pd.read_csv("F:\File_Kuliah\data\image_class_labels.txt",
    ...:                         sep=' ', header=None, names=['imgid', 'label'])
    ...: imglabels = imglabels.set_index('imgid')
    ...: imglabels.head ()
    ...: imglabels.shape
Out[26]: (11788, 1)
```

Figure 3.9: Membaca Dataset label

5. Selanjutnya data imgatt2 akan join dengan imglabels atau menggabungkan field dari dua file yang terpisah, karena data baris sama banyak nya maka ditambahkan data kolom dari imgatt2 312 kolom dengan imglabels 1 kolom, maka hasilnya menjadi 313 kolom

```
In [27]:
    ...: df = imgatt2.join (imglabels)
    ...: df = df.sample(frac=1)
```

Figure 3.10: Menggabungkan field dari file yang terpisah

6. Kemudian memisahkan dan memilih label atau memecah data kembali seperti sebelumnya dengan cara mengambil 312 dari belakang dan mengambil 312 dari depan.

```
In [28]:
    ...: df_att = df.iloc[:, :312]
    ...: df_label = df.iloc[:, 312:]
```

Figure 3.11: Memisahkan dan memilih label

7. Membagi Data training dan data testing dengan mengambil row sebanyak 8000 dari akhir untuk training dan 8000 dari awal untuk data testing.

```
In [29]:
    ...: df_train_att = df_att [:8000]
    ...: df_train_label = df_label [:8000]
    ...: df_test_att = df_att [8000:]
    ...: df_test_label = df_label [8000:]
    ...:
    ...: df_train_label = df_train_label ['label']
    ...: df_test_label = df_test_label ['label']
```

Figure 3.12: Pembagian data training dan tes

8. Melakukan klasifikasi, di dalam kelas randomforestclassifier setting parameter variable yaitu 50 dalam satu independen tree maksimal akan mengakomodir 50 atribut. Kemudian lakukan instansiasi dengan melakukan klasifikasi pada data train att beserta data label nya.

```
In [32]:
    ...: from sklearn.ensemble import RandomForestClassifier
    ...: clf = RandomForestClassifier (max_features=50, random_state=0, n_estimators=100)
    ...:
    ...: # Fitting random forest dengan dataset training
    ...: clf.fit (df_train_att , df_train_label)
Out[32]: RandomForestClassifier(max_features=50, random_state=0)
```

Figure 3.13: Instansiasi kelas Random Forest

9. kode dibawah ini digunakan untuk mengetahui hasil prediksi dan menghitung score perolehan dari klasifikasi.

```
In [33]:
    ...: print (clf.predict(df_train_att.head()))
    ...:
    ...: # Score perolehan dari klasifikasi
    ...: clf.score(df_test_att, df_test_label)
[ 32  28   3 181 107]
Out[33]: 0.4477296726504752
```

Figure 3.14: Hasil Prediksi dan score klasifikasi

### 3.2.5 Program Confusion Matrix

Berikut merupakan output dari percobaan Confusion Matrix yang telah dilakukan :

1. Memetakan Random Forest ke dalam Confusion matrix, dibawah ini import terlebih dahulu library confusion matrix nya.

```
In [36]: from sklearn.metrics import confusion_matrix
    ...: pred_labels = clf.predict(df_test_att)
    ...: cm = confusion_matrix ( df_test_label, pred_labels )
    ...: cm
Out[36]:
array([[ 4,  0,  1, ...,  0,  0,  0],
       [ 1, 11,  0, ...,  0,  1,  0],
       [ 4,  0,  9, ...,  0,  0,  0],
       ...,
       [ 0,  0,  1, ...,  2,  0,  0],
       [ 0,  0,  0, ...,  0,  8,  0],
       [ 0,  0,  0, ...,  0,  0, 14]], dtype=int64)
```

Figure 3.15: Memetakan Random Forest ke dalam Confusion matrix

```
In [39]: 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.
    ...:     #for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    ...:     #    plt.text(j, i, format(cm[i, j], fmt),
    ...:     #             horizontalalignment="center",
    ...:     #             color="white" if cm[i, j] > thresh else "black")
    ...:
    ...:     plt.tight_layout()
    ...:     plt.ylabel('True label')
    ...:     plt.xlabel('Predicted label')
```

Figure 3.16: Plotting Confusion Matrix

2. Selanjutnya plotting confusion matrix menggunakan matplotlib.

3. Membaca File Classes

```
In [37]: birds = pd.read_csv("F:\File_Kuliah\data\classes.txt",
    ...:                     sep='\s+', header=None, usecols=[1], names=['birdname'])
    ...: birds = birds['birdname']
    ...: birds
Out[37]:
0      001.Black_footed_Albatross
1           002.Laysan_Albatross
2           003.Sooty_Albatross
3         004.Groove_billed_Ani
4            005.Crested_Auklet
                 ...
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 3.17: Membaca file classes

4. Plot hasil perubahan label

```
In [40]: 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
[[0.25 0.   0.06 ... 0.   0.   0.  ]
 [0.05 0.58 0.   ... 0.   0.05 0.  ]
 [0.17 0.   0.38 ... 0.   0.   0.  ]
 ...
 [0.   0.   0.04 ... 0.07 0.   0.  ]
 [0.   0.   0.   ... 0.   0.5  0.  ]
 [0.   0.   0.   ... 0.   0.   0.82]]
```

Figure 3.18: Plot hasil perubahan label

### 3.2.6 Program Klasifikasi SVM dan Decission Tree

Berikut merupakan output dari percobaan Klasifikasi SVM dan Decission Tree yang telah dilakukan :

1. Klasifikasi decission tree dengan menggunakan dataset yang sama

```
In [25]: from sklearn import tree
    ...: clftree = tree.DecisionTreeClassifier()
    ...: clftree.fit(df_train_att, df_train_label)
    ...: clftree.score(df_test_att, df_test_label)
Out[25]: 0.2624076029567054
```

Figure 3.19: Klasifikasi Decission Tree

2. Klasifikasi SVM dengan menggunakan dataset yang sama

```
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)
Out[26]: 0.4675290390707497
```

Figure 3.20: Klasifikasi SVM

### 3.2.7 Program Cross Validation

Berikut merupakan output dari percobaan Program Cross Validation yang telah dilakukan :

1. Hasil Cross Validation untuk Random forest

```
In [41]: 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: 0.34 (+/- 0.02)
```

Figure 3.21: Hasil Cross Validation Random Forest

2. Hasil Cross Validation untuk Decission Tree

```
In [28]: 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 (+/- 0.01)
```

Figure 3.22: Hasil Cross Validation Decission Tree

3. Hasil Cross Validation untuk SVM

### 3.2.8 Program Pengamatan Komponen Informasi

Berikut merupakan output dari percobaan Program Pengamatan Komponen Informasi yang telah dilakukan :

1. Ouput berikut dapat mengetahui banyaknya tree yang dibuat, berapa atribut yang digunakan dan informasi lainnya.

2. Output berikut merupakan hasil plotting komponen informasi agar dapat dibaca

```
In [29]: scoressvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv=5)
    ...: print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(), scoressvm.std() * 2))
Accuracy: 0.47 (+/- 0.02)
```

Figure 3.23: Hasil Cross Validation 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: %0.2f (+/- %0.2f)" %
    ...:               (max_features, n_estimators, scores.mean(), scores.std() * 2))
Max features: 5, num estimators: 10, accuracy: 0.27 (+/- 0.02)
Max features: 5, num estimators: 30, accuracy: 0.36 (+/- 0.01)
Max features: 5, num estimators: 50, accuracy: 0.39 (+/- 0.01)
Max features: 5, num estimators: 70, accuracy: 0.41 (+/- 0.00)
Max features: 5, num estimators: 90, accuracy: 0.42 (+/- 0.02)
Max features: 5, num estimators: 110, accuracy: 0.43 (+/- 0.01)
```

Figure 3.24: Hasil plotting komponen

```
In [37]: import matplotlib.pyplot as plt
    ...: from mpl_toolkits.mplot3d import Axes3D
    ...: from matplotlib import cm
    ...: fig = plt.figure()
    ...: fig.clf()
    ...: ax = fig.add_subplot(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()
```

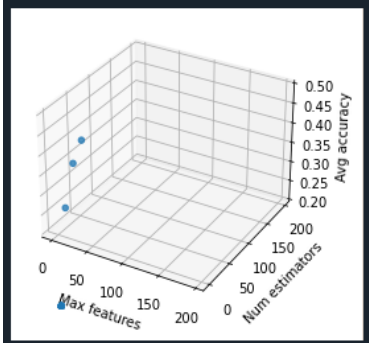


Figure 3.25: Hasil plotting komponen