IMPLEMENTASI DATA MINING PADA *DATASET* BREAST CANCER WISCONSIN

2018

Dosen Kelas :
Dr. Imam Mukhlash, MT

Disusun Oleh:
Venansius Ryan Tjahjono 06111540000043
Sumihar Christian N. S. 06111540000115

DECEMBER 17

DEPARTEMEN MATEMATIKA
FAKULTAS MATEMATIKA, KOMPUTASI, DAN SAINS DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa karena atas berkat dan karunia-Nya, tugas besar Data Mining ini bisa diselesaikan tepat waktu.

Semakin berkembangnya dunia medis, tak luput dengan perkembangan dari penyakit, terutama penyakit kanker payudara. Penyakit ini merupakan penyakit mematikan bagi kaum hawa maupun kaum adam. Berjalan dari hal ini, penulis menganalisis *dataset* kanker payudara untuk mendapatkan informasi yang bisa digunakan untuk membantu prediksi apakah kanker payudara yang dialami seseorang adalah kanker ganas atau kanker jinak. Selain itu, diharap bisa menjadi suatu cara untuk menganalisis penyakit kanker payudara secara umum pada rumah sakit atau pun di klinik kesehatan.

Penulis juga berterima kasih, khususnya kepada, Dr. Imam Mukhlash, S.Si, M.T, selaku dosen kelas Data Mining, yang telah memberikan ilmu berupa pengalaman-pengalaman dan materi kepada kami. Tak lupa juga, penulis berterima kasih kepada teman-teman kelas Data Mining yang sudah saling membantu dalam menyelesaikan tugas besar ini.

Akhir kata, penulis sadar bahwa tulisan ini mempunyai kekurangan. Oleh karena itu, harap dimaklumi. Sekian dan terima kasih.

Surabaya, 12 Desember 2018

Penulis

DAFTAR ISI

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Data *mining* merupakan serangkaian proses untuk mendapatkan informasi yang berguna dari gudang basis data yang besar. Data *mining* juga dapat diartikan sebagai pengekstrakan informasi baru yang diambil dari bongkahan data besar yang membantu dalam pengambilan keputusan. Dalam data *mining* terdapat banyak teknik dalam pengerjaannya, diantaranya yaitu algoritma *naïve bayes, decision tree*, kNN, SVM, jaringan saraf tiruan dan masih banyak lainnya.

Teknik *data mining* secara garis besar dapat dibagi menjadi dua kelompok, yaitu verifikasi dan *discovery*. Metode verifikasi umumnya meliputi teknik – teknik statistik seperti *goodness of fit* dan analisis variansi. Metode *discovery* sendiri dapat terdapat dua model yaitu model prediktif dan model deskriptif. Teknik prediktif melakukan prediksi terhadap data dengan menggunakan hasil – hasil yang telah diketahui dari data yang berbeda. Model ini dapat digunakan dengan data historis lain. Sementara itu teknik deskriptif bertujuan untuk mengidentifikasi pola – pola atau hubungan antar data dan memberikan cara untuk mengeksplorasi karakteristik data yang diselidiki.

Data mining sendiri memiliki beberapa teknik untuk menemukan pola atau informasi yang tersembunyi salah satunya yang banyak digunakan dalam 2 penelitian kebanyakan adalah teknik *cluster*. Teknik *cluster* sendiri merupakan teknik yang tidak menggunakan parameter atau disebut juga *non parametric* dan diaplikasikan untuk kasus nyata. Untuk dapat mengaplikasikan teknik ini perlu adanya algoritma yang bekerja, *data mining* juga memiliki beberapa algoritma tetapi yang paling sederhana dan sering digunakan adalah algoritma *k-means*, algoritma ini sendiri bertujuan untuk mengelompokan obyek ke dalam *cluster* atau kelompok yang telah

ditentukan. *Data mining* mulai banyak digunakan sebagai bahan penelitian dalam hal mencari pola atau nilai dari suatu basis data yang besar.

Kanker payudara merupakan jenis kanker yang menempati urutan kedua sebagai penyakit yang paling umum d*item*ui. Seperlima dari wanita penderita kanker adalah mereka yang didiagnosa mengidap kanker payudara. Kanker secara umum dibagi dua yaitu jinak dan ganas, begitupun kanker payudara. Pada status ganas, kanker dapat berakibat buruk bagi penderitanya bila terlambat diketahui. *World Health Organization* (2016) menyebutkan kanker payudara adalah kanker paling umum terjadi pada wanita baik di negara maju dan berkembang. Dokter Spesialis Bedah Payudara Rumah Sakit Onkologi Surabaya, dr. Dwirani Rosmala, Sp.B juga menyebut, setiap tahun muncul 500 hingga 600 pasien kanker payudara baru. Meskipun kanker payudara dianggap penyakit dari negara maju, hampir 50% dari kasus kanker payudara dan 58% kematian terjadi di negara-negara kurang berkembang. Seperti wanita, pria memiliki jaringan payudara, meskipun dalam jumlah yang lebih kecil. Ini berarti bahwa pria juga dapat terkena kanker payudara, meskipun tidak banyak. Risiko seorang pria didiagnosa menderita kanker payudara sebelum usia 75 tahun adalah satu dari 1258 orang. Sedangkan, risiko seorang wanita didiagnosa menderita kanker payudara sebelum usia 85 tahun adalah satu dari delapan orang (*Breast Cancer Network Australia*, 2014).

Oleh karena itu, penulis memberikan sebuah kajian mengenai analisis *dataset* kanker payudara yang diharapkan bisa menambah informasi untuk mengetahui penyakit kanker payudara (jinak atau ganas).

1.2. Rumusan Masalah

Dari latar belakang yang diperoleh, dirumuskan beberapa rumusan masalah yang akan dibahas pada tulisan ini.

1. Bagaimana cara *preprocessing data* agar mendapatkan data yang siap diolah?

2. Apa hasil analisis dari *dataset* kanker payudara yang diperoleh dengan menggunakan *task data mining*?
3. Bagaimana cara melakukan prediksi kemungkinan kanker dengan melakukan *cross-validation* pada *dataset* kanker payudara dengan menggunakan *Multi-Layer Perceptron.*

1.3. Tujuan Penulisan

1. Mengetahui cara *preprocessing data* agar mendapatkan data yang siap diolah.
2. Mendapatkan hasil analisis dari *dataset* kanker payudara yang diperoleh dengan menggunakan *task data mining*.
3. Memberikan prediksi kemungkinan kanker dengan melakukan *cross-validation* pada *dataset* kanker payudara dengan menggunakan *Multi-Layer Perceptron.*

1.4. Manfaat Penulisan

4. Memberikan informasi mengenai cara *preprocessing data* agar mendapatkan data yang siap diolah.
5. Memberikan kontribusi ilmu mengenai hasil analisis dari *dataset* kanker payudara yang diperoleh dengan menggunakan *task data mining*.
6. Memberikan prediksi kemungkinan kanker dengan melakukan *cross-validation* pada *dataset* kanker payudara dengan menggunakan *Multi-Layer Perceptron.*

1.5. Perangkat Lunak yang Digunakan

1. Python
2. R
3. WEKA

BAB 2
TINJAUAN PUSTAKA

Pada bab ini, akan dijelaskan mengenai beberapa algoritma yang akan digunakan penulis untuk menyelesaikan permasalahan yang ada.

2.1 Algoritma *k-nearest neighbor*

Algoritma *k-nearest neighbor* (k-NN atau KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Data pembelajaran diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi data pembelajaran. Sebuah titik pada ruang ini ditandai kelas *c* jika kelas *c* merupakan klasifikasi yang paling banyak di*item*ui pada *k* buah tetangga terdekat titk tersebut. Dekat atau jauhnya tetangga biasanya dihitung berdasarkan jarak Euclidean. Pada fase pembelajaran, algoritma ini hanya melakukan penyimpanan vektor-vektor fitur dan klasifikasi dari data pembelajaran.

Pada fase klasifikasi, fitur-fitur yang sama dihitung untuk data test (yang klasifikasinya tidak diketahui). Jarak dari vektor yang baru ini terhadap seluruh vektor data pembelajaran dihitung, dan sejumlah *k* buah yang paling dekat diambil. Titik yang baru klasifikasinya diprediksikan termasuk pada klasifikasi terbanyak dari titik-titik tersebut. Nilai *k* yang terbaik untuk algoritma ini tergantung pada data; secara umumnya, nilai *k* yang tinggi akan mengurangi efek *noise* pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi lebih kabur. Nilai *k* yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan cross-validation. Kasus khusus di mana klasifikasi diprediksikan berdasarkan data pembelajaran yang paling dekat (dengan kata lain, $k = 1$) disebut algoritma *nearest neighbor*.

2.2 Algoritma Naïve Bayes

Naive Bayes merupakan salah satu algoritma bagian dari metode Machine Learning yang proses pengerjaannya menggunakan perhitungan probabilitas. Adapun konsep dasar yang digunakan algoritma Naive Bayes ini yaitu Teorema Bayes, teorema di dalam statistika guna untuk menghitung peluang. Teorema Bayes ini digunakan untuk melakukan proses perhitungan probabilitas terjadinya peristiwa dengan berdasar kepada pengaruh – pengaruh yang didapat dari sebuah hasil suatu observasi. Selanjutnya, penerapan Algoritma Naive Bayes banyak digunakan pada contoh-contoh berikut.

1. Klasifikasi Kendaraan Roda Dua Menggunakan Metode Naive Bayes.
2. Klasifikasi dan *Cluster*ing Penjurusan Siswa SMK N1 Kandis Menggunakan Algoritma Naive Bayes.
3. Klasifikasi Sagu Dengan Menggunakan Algoritma Naive Bayes Classification. (Studi Kasus Dinas Kehutanan Pekanbaru, Riau).
4. Penerapan Algoritma Naive Bayes Dalam Mengklasifikasi Pola Kelulusan Mahasiswa Universitas Gajah Mada. (Studi Kasus: Fakultas Teknik Informatika).
5. Memprediksi Masa Studi Mahasiswa Sistem Informasi S1 Fakultas Ilmu Komputer Universitas Sultan Syarif Qasim, Pekanbaru menggunakan Algoritma Naive Bayes.

2.3 *Support Vector Machine*

Menurut Santoso (2007) *Support* vector machine (SVM) adalah suatu teknik untuk melakukan prediksi, baik dalam kasus klasifikasi maupun regresi. SVM berada dalam satu kelas dengan Artificial Neural Network (ANN) dalam hal fungsi dan kondisi permasalahan yang bisa diselesaikan. Keduanya masuk dalam kelas *supervised learning*.

Pada permasalahan yang kompleks atau permasalahan dengan parameter yang banyak, metode ini sangat baik untuk digunakan. Metode ini juga baik digunakan untuk mendiagnosis berbagai macam jenis penyakit. Salah satu kelebihan yang dimiliki metode SVM adalah penanganan *error* pada set data

training yang menggunakan *Structural Risk Minimization* (SRM). SRM dikatakan lebih baik karena tidak hanya meminimalkan *error* yang terjadi, tetapi meminimalkan faktor-faktor lainnya.

2.4 Algoritma *k-Means*

K-means merupakan salah satu algoritma *clustering*. Tujuan algoritma ini yaitu untuk membagi data menjadi beberapa kelompok. Algoritma ini menerima masukan berupa data tanpa label kelas. Hal ini berbeda dengan *supervised learning* yang menerima masukan berupa vektor (x_1 , y_1) , (x_2 , y_2) , …, (x_i , y_i), di mana $x_i$merupakan data dari suatu data pelatihan dan y_i merupakan label kelas untuk x_i.

Pada algoritma pembelajaran ini, komputer mengelompokkan sendiri data-data yang menjadi masukannya tanpa mengetahui terlebih dulu target kelasnya. Pembelajaran ini termasuk dalam *unsupervised learning.* Masukan yang diterima adalah data atau objek dan *k* buah kelompok (*cluster*) yang diinginkan. Algoritma ini akan mengelompokkan data atau objek ke dalam *k* buah kelompok tersebut. Pada setiap *cluster* terdapat titik pusat (*centroid*) yang merepresentasikan *cluster* tersebut. Berikut adalah tahapan dari algoritma *k-means.*

1. Pilih K buah titik *centroid* secara acak
2. Kelompokkan data sehingga terbentuk K buah *cluster* dengan titik *centroid* dari setiap *cluster* merupakan titik *centroid* yang telah dipilih sebelumnya
3. Perbaharui nilai titik *centroid*
4. Ulangi langkah 2 dan 3 sampai nilai dari titik *centroid* tidak lagi berubah

Proses pengelompokkan data ke dalam suatu *cluster* dapat dilakukan dengan cara menghitung jarak terdekat dari suatu data ke sebuah titik *centroid.* Perhitungan jarak Minkowski dapat digunakan untuk menghitung jarak antar 2 buah data. Rumus untuk menghitung jarak tersebut adalah

$$d(x_i, x_j) = (|x_{i1} - x_{j1}|^g + |x_{i2} - x_{j2}|^g + \ldots + |x_{ip} - x_{jp}|^g)^{1/g}$$

2.5 *Spectral Clustering*

Dalam statistik multivariat dan pengelompokan data, teknik pengelompokan spektral memanfaatkan spektrum (nilai eigen) dari matriks kesamaan data untuk melakukan pengurangan dimensi sebelum mengelompokkan dalam dimensi yang lebih sedikit. Matriks kesamaan disediakan sebagai masukan dan terdiri dari penilaian kuantitatif dari kemiripan relatif masing-masing pasangan titik dalam *dataset*. Dalam aplikasi untuk segmentasi gambar, pengelompokan spektral dikenal sebagai pengelompokan objek berbasis segmentasi.

2.6 *Hierarchical Clustering*

Metode ini memulai pengelompokan dengan dengan dua atau lebih objek yang mempunyai kesamaan paling dekat. Kemudian proses diteruskan ke objek lain yang mempunyai kedekatan kedua. Demikian seterusnya sehingga cluster akan membentuk semacam "pohon", dimana ada hirarki (tingkatan) yang jelas antar objek, dari yang paling mirip sampai paling tidak mirip. Secara logika semua objek pada akhirnya akan membentuk sebuah cluster. Dendogram biasanya digunakan untuk membantu memperjelas proses hirarki tersebut. Dalam metode hirarki cluster terdapat dua tipe dasar yaitu *agglomerative* (pemusatan) dan divisive (penyebaran). Dalam metode agglomerative, setiap obyek atau observasi dianggap sebagai sebuah cluster tersendiri. Dalam tahap selanjutnya, dua cluster yang mempunyai kemiripan digabungkan menjadi sebuah cluster baru demikian seterusnya. Sebaliknya, dalam metode *divisive* kita beranjak dari sebuah cluster besar yang terdiri dari semua obyek atau observasi. Selanjutnya, obyek atau observasi yang paling tinggi nilai ketidakmiripannya kita pisahkan demikian seterusnya. Algoritma ini dituliskan sebagai berikut.

1. Hitung matrik jarak antar data.
2. Gabungkan dua kelompok terdekat berdasarkan parameter kedekatan yang ditentukan.

3. Perbarui matrik jarak antar data untuk merepresentasikan kedekatan diantara kelompok baru dan kelompok yang masih tersisa.
4. Ulangi langkah 2 dan 3 higga hanya satu kelompok yang tersisa.

2.7 Algoritma Apriori

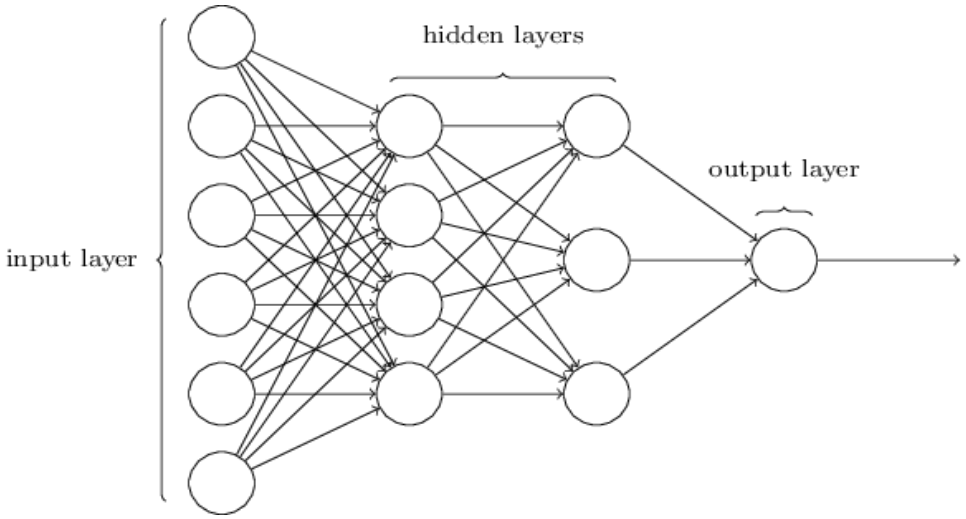
Algoritma Apriori adalah algoritma paling terkenal untuk menemukan pola frekuensi tinggi. Pola frekuensi tinggi adalah pola-pola *item* di dalam suatu *database* yang memiliki frekuensi atau *support* di atas ambang batas tertentu yang disebut dengan istilah *minimum support*. Pola frekuensi tinggi ini digunakan untuk menyusun aturan assosiatif dan juga beberapa teknik data mining lainnya. Walaupun akhir-akhir ini dikembangkan banyak algoritma yang lebih efisien dari Apriori seperti FP-growth, LCM dsb, tetapi Apriori tetap menjadi algoritma yang paling banyak diimplementasikan dalam produk komersial untuk data mining karena dianggap algoritma yang paling mapan. Algoritma Apriori dibagi menjadi beberapa tahap yang disebut iterasi atau pass. Tiap iterasi menghasilkan pola frekuensi tinggi dengan panjang yang sama dimulai dari pass pertama yang menghasilkan pola frekuensi tinggi dengan panjang satu. Di iterasi pertama ini, *support* dari setiap *item* dihitung dengan melakukan scan *database*. Setelah *support* dari setiap *item* didapat, *item* yang memiliki *support* diatas *minimum support* dipilih sebagai pola frekuensi tinggi dengan panjang 1 atau sering disingkat 1-*item*set. Singkatan k-*item*set berarti satu set yang terdiri dari k *item*.

2.8 *Prefix Span*

Prefix Span adalah salah satu metode penggalian pola sekuensial yang menggunakan pendekatan pattern growth serta melakukan proyeksi sufiks terhadap basis data sekuens, sedangkan *AprioriAll* adalah salah satu metode penggalian pola sekuensial yang menggunakan pendekatan Apriori serta melakukan pembangkitan dan pengujian sekuens kandidat.

2.9 Multi-Layer Perceptron

Multi-layer perceptron (MLP) adalah pengembangan dari model *perceptron* yang dikembangkan oleh Rosenblatt pada tahun 1958.



Perceptron memiliki keterbatasan hanya dapat menyelesaikan masalah-masalah yang linier, sedangkan MLP dapat digunakan untuk menyelesaikan masalah yang lebih kompleks. Pada dasarnya, MLP adalah *perceptron* yang memiliki *layer* atau lapisan tambahan diantara *layer* input (neuron $\boldsymbol{X_i}$) dan layer output (neuron $\boldsymbol{Y_i}$) yang disebut dengan *hidden layer*. Proses perhitungan dari setiap neuronnya sama dengan *perceptron*. Sinyal output neuron ($\boldsymbol{v}$) dimasukkan kedalam sebuah fungsi aktivasi. (Fausett, 2006)(Ham & Kostanic, 2001).

BAB 3
PENJELASAN DATA SET

3.1 Sumber Data Set

Dataset diperoleh penulis dari *https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/.* Data bisa juga diperoleh dari UCI *Machine Learning Repository* atau dari *kaggle.com*. Data terdiri atas 32 atribut dan 549 *entry.*

3.2 Penjelasan Atribut Data Set

Berikut adalah penjelasan berupa *summary* (berisi Min, Kuartil, Max) atribut dari *dataset* yang digunakan yang memanfaatkan perangkat lunak R. Pertama, kita definisikan *database* kita adalah sebagai

```
bc <- read.csv("D:/ITS/7th/Data Mining/Tugas 5 Big Project/data.csv",
header=TRUE)
```

1. Id : berisi data ID *number.*

```
> summary(bc$id)
      Min.    1st Qu.     Median       Mean    3rd Qu.    Max.
     8670     869218     906024   30371831    8813129
911320502
```

2. Diagnosis : Hasil diagnosis lapisan payudara.

```
> summary(bc$diagnosis)
  Benign Malignant
  357       212
```

3. Radius_mean : rataan jarak pusat ke titik ujung dari payudara.

```
> summary(bc$radius_mean)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  6.981  11.700  13.370  14.127  15.780  28.110
```

4. Texture_mean : standar deviasi dari nilai *gray-scale.*

```
> summary(bc$texture_mean)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  9.71   16.17   18.84   19.29   21.80   39.28
```

5. Perimeter_mean : rataan ukuran inti tumor.

```
> summary(bc$perimeter_mean)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  43.79   75.17   86.24   91.97  104.10  188.50
```

6. Area_mean : rataan luasan dari tumor.

```
> summary(bc$area_mean)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  143.5   420.3   551.1   654.9   782.7  2501.0
```

7. Smoothness_mean : rataan variasi lokal dari panjang jari-jari.

```
> summary(bc$smoothness_mean)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.05263 0.08637 0.09587 0.09636 0.10530 0.16340
```

8. Compactness_mean : rataan dari keliling^2/luasan -1

```
> summary(bc$compactness_mean)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01938 0.06492 0.09263 0.10434 0.13040 0.34540
```

9. Concavity_mean : rataan dari *severity* dari cekungan kontur tumor.

```
> summary(bc$concavity_mean)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00000 0.02956 0.06154 0.08880 0.13070 0.42680
```

10. Concave points_mean : rataan dari banyaknya cekungan kontur tumor.

```
> summary(bc$concave.points_mean)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00000 0.02031 0.03350 0.04892 0.07400 0.20120
```

11. Symmetry_mean : rataan simetri.

```
> summary(bc$symmetry_mean)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1060  0.1619  0.1792  0.1812  0.1957  0.3040
```

12. fractal_dimension_mean : rataan dari keliling tumor – 1.

```
> summary(bc$fractal_dimension_mean)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.04996 0.05770 0.06154 0.06280 0.06612 0.09744
```

13. Radius_se : standard error dari radius_mean.

```
> summary(bc$radius_se)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1115  0.2324  0.3242  0.4052  0.4789  2.8730
```

14. Texture_se : standard error dari standar deviasi *gray-scale*.

```
> summary(bc$texture_se)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.3602  0.8339  1.1080  1.2169  1.4740  4.8850
```

15. Perimeter_se : standard error dari perimeter_mean.

```
> summary(bc$perimeter_se)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.757   1.606   2.287   2.866   3.357  21.980
```

16. Area_se : standard error dari area_mean.

```
> summary(bc$area_se)
   Min. 1st Qu.  Median    Mean 3rd Qu.  Max.
  6.802  17.850  24.530  40.337  45.190  542.200
```

17. Smoothness_se : standard error dari smoothness_mean.

```
> summary(bc$smoothness_se)
    Min.  1st Qu.   Median     Mean  3rd Qu.  Max
0.001713 0.005169 0.006380 0.007041 0.008146  0.031130
```

18. Compactness_se : standard error dari compactness_mean.

```
> summary(bc$compactness_se)
    Min.  1st Qu.   Median     Mean  3rd Qu.  Max
0.002252 0.013080 0.020450 0.025478 0.032450  0.135400
```

19. Concavity_se : standard error dari concavity_mean.

```
> summary(bc$concavity_se)
   Min. 1st Qu.  Median    Mean 3rd Qu.  Max.
0.00000 0.01509 0.02589 0.03189 0.04205  0.39600
```

20. Concave points_se : standard error dari concave_points_mean.

```
> summary(bc$concave.points_se)
    Min.  1st Qu.   Median     Mean  3rd Qu.  Max
0.000000 0.007638 0.010930 0.011796 0.014710  0.052790
```

21. Symmetry_se : standard error dari symmetry_mean.

```
> summary(bc$symmetry_se)
    Min.  1st Qu.   Median     Mean  3rd Qu.  Max
0.007882 0.015160 0.018730 0.020542 0.023480  0.078950
```

22. fractal_dimension_se : standard error dari fractal_dimension_mean.

```
> summary(bc$fractal_dimension_se)
     Min.   1st Qu.    Median      Mean   3rd Qu.    Max.
0.0008948 0.0022480 0.0031870 0.0037949 0.0045580
0.0298400
```

23. Radius_worst : nilai rataan terbesar (*worst*) dari radius_mean.

```
> summary(bc$radius_worst)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   7.93   13.01   14.97   16.27   18.79   36.04
```

24. Texture_worst : nilai rataan terbesar (*worst*) dari texture_mean.

```
> summary(bc$texture_worst)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12.02   21.08   25.41   25.68   29.72   49.54
```

25. Perimeter_worst : nilai rataan terbesar (*worst*) dari perimeter_mean.

```
> summary(bc$perimeter_worst)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  50.41   84.11   97.66  107.26  125.40  251.20
```

26. Area_worst : nilai rataan terbesar (*worst*) dari area_mean.

```
> summary(bc$area_worst)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  185.2   515.3   686.5   880.6  1084.0  4254.0
```

27. Smoothness_worst : nilai rataan terbesar (*worst*) dari smoothness_worst.

```
> summary(bc$smoothness_worst)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.07117 0.11660 0.13130 0.13237 0.14600 0.22260
```

28. Compactness_worst : nilai rataan terbesar (*worst*) dari compactness_mean.

```
> summary(bc$compactness_worst)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.02729 0.14720 0.21190 0.25427 0.33910 1.05800
```

29. Concavity_worst : nilai rataan terbesar (*worst*) dari concavity_mean.

```
> summary(bc$concavity_worst)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.1145  0.2267  0.2722  0.3829  1.2520
```

30. Concave points_worst : nilai rataan terbesar (*worst*) dari concave_points_mean.

```
> summary(bc$concave.points_worst)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00000 0.06493 0.09993 0.11461 0.16140 0.29100
```

31. Symmetry_worst : nilai rataan terbesar (*worst*) dari symmetry_mean.

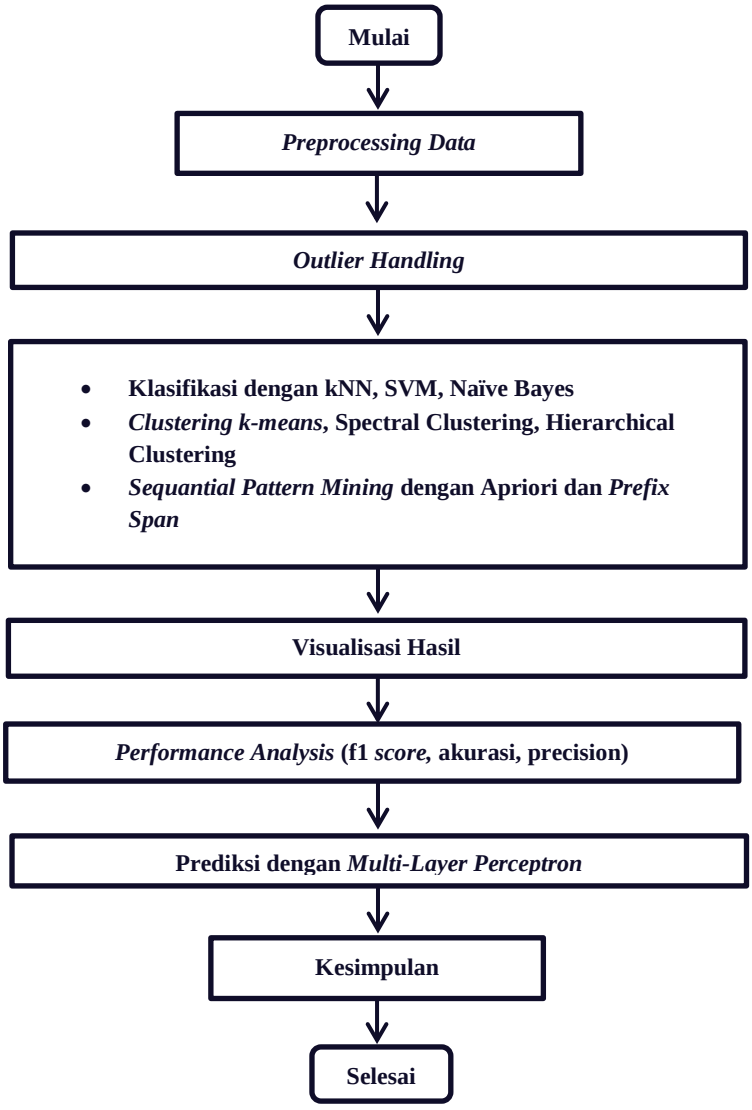
```
> summary(bc$symmetry_worst)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1565  0.2504  0.2822  0.2901  0.3179  0.6638
```

32. fractal_dimension_worst : nilai rataan terbesar (*worst*) dari fractal_dimension_mean.

```
> summary(bc$fractal_dimension_worst)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.05504 0.07146 0.08004 0.08395 0.09208 0.20750
```

BAB 4
METODOLOGI

Pada bagian ini akan dijelaskan mengenai langkah-langkah yang digunakan dalam penyelesaian masalah pada tulisan ini. Selain itu, dijelaskan pula prosedur dan proses tiap-tiap langkah yang dilakukan dalam penyelesaian rumusan masalah. Adapun langkah-langkah sistematis yang dilakukan dalam proses pengerjaan tulisan ini, yaitu sebagai berikut:



BAB 5
PEMBAHASAN

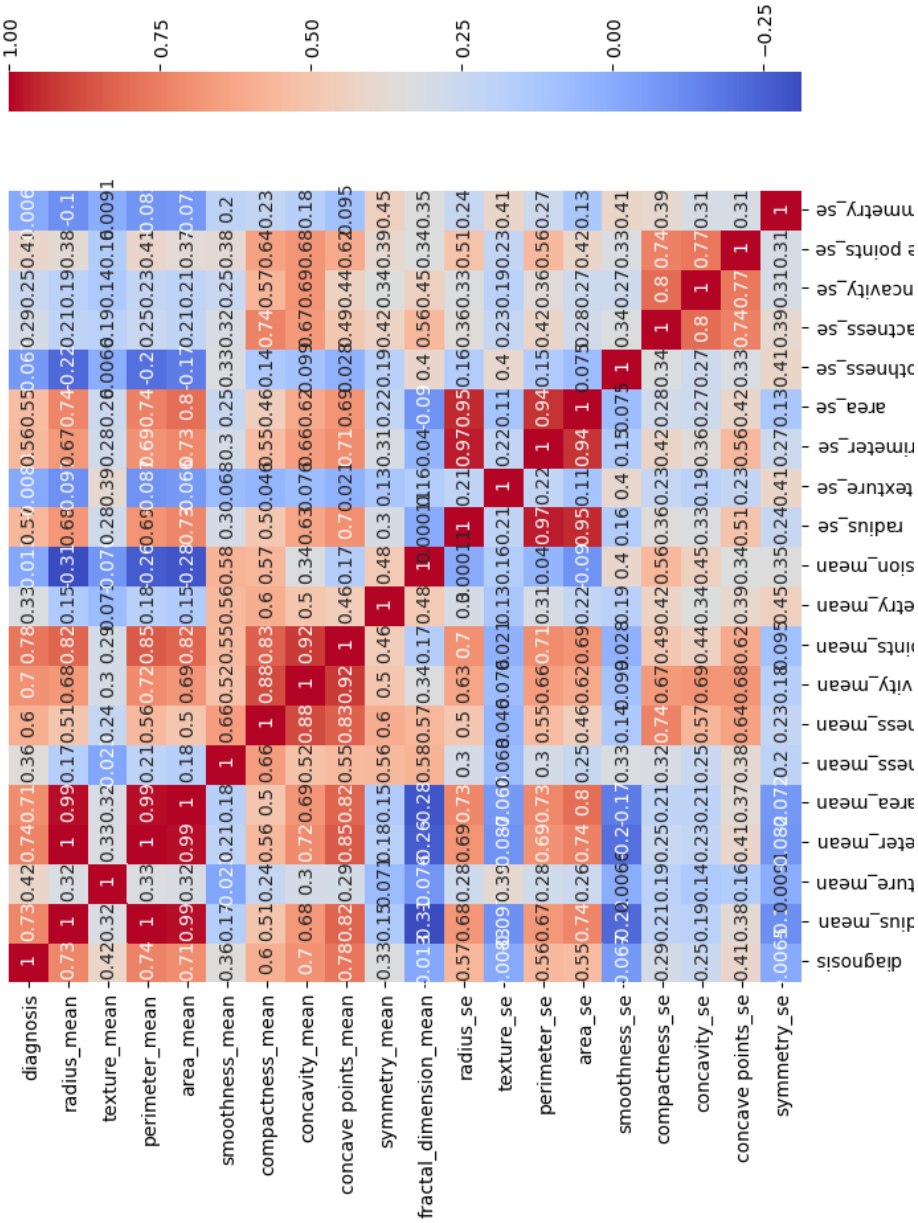
5.1 Plot Data Set

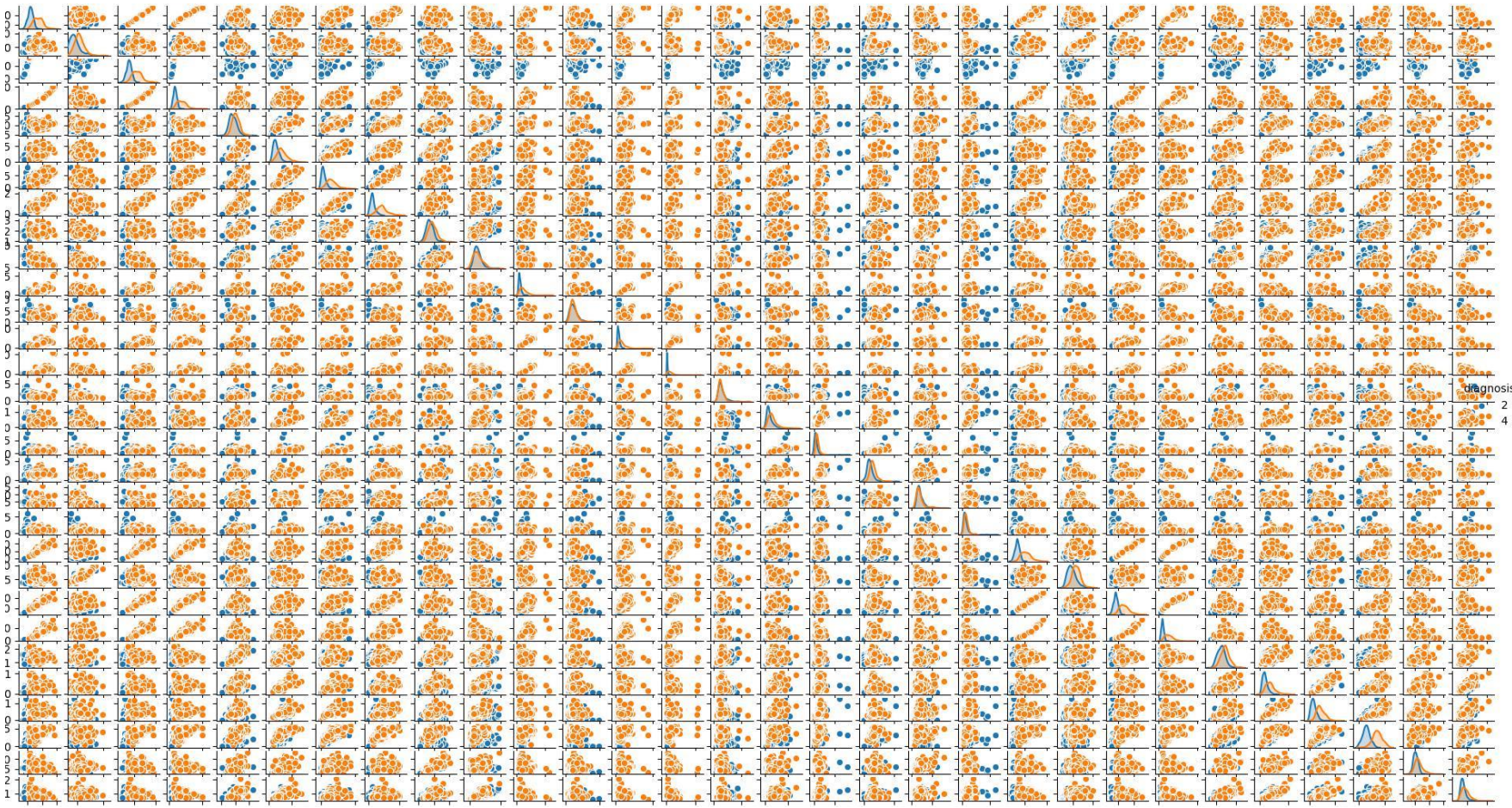
Berikut dilakukan beberapa plot dari atribut pada data yang digunakan dengan menggunakan perangkat lunak R dan Python. Pada gambar dibawah ini disajikan visualisasi data awal dengan menggunakan *Exploratory Data Analysis* (EDA) pada Python terlebih dahulu. Diperoleh atribut yang paling berpengaruh adalah radius, perimeter, dan area.

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split

# Importing the dataset
dataset = pd.read_csv('data.csv', index_col=0)
print(dataset.head())
X = dataset.iloc[:, 1:31].values # 11 karena ada class
y = dataset.iloc[:, 0].values # 10 karena tidak ada kelas
mapping={'M':4, 'B':2}
print(dataset.shape)
dataset['diagnosis'] = dataset['diagnosis'].map(mapping)

print("\n \t The data frame has {0[0]} rows and {0[1]} columns.
\n".format(dataset.shape))
dataset.info()

print(dataset.head(3))

#visualizing data
features_mean = list(dataset.columns[0:20])
print(features_mean)
# plt.figure(figsize=(32,32))
sns.heatmap(dataset[features_mean].corr(), annot=True,
square=True, cmap='coolwarm')
plt.show()

print(dataset.columns)
```

```
sns.pairplot(dataset, hue='diagnosis', vars =
["radius_mean","texture_mean","perimeter_mean","area_mean","smoot
hness_mean","compactness_mean","concavity_mean","concave
points_mean","symmetry_mean","fractal_dimension_mean","radius_se"
,"texture_se","perimeter_se","area_se","smoothness_se","compactne
ss_se","concavity_se","concave
points_se","symmetry_se","fractal_dimension_se","radius_worst","t
exture_worst","perimeter_worst","area_worst","smoothness_worst","
compactness_worst","concavity_worst","concave
points_worst","symmetry_worst","fractal_dimension_worst"])
plt.show()

#Print benign and malign cancer

sns.countplot(dataset['class'], label = "Hitung")
plt.show()

# # Feature Scaling
# from sklearn.preprocessing import StandardScaler
# sc = StandardScaler()
# X_train = sc.fit_transform(X_train)
# X_test = sc.transform(X_test)

X = dataset.drop(['class'], axis = 1) # We drop our "target"
feature and use all the remaining features in our dataframe to
train the model.
print(X.head())
y = dataset['class']
print('\n')
print(y.head())

# Splitting the dataset into the Training set and Test set
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.20)

# check the test and train data
print ('The size of our training "X" (input features) is',
X_train.shape)
print ('\n')
print ('The size of our testing "X" (input features) is',
X_test.shape)
print ('\n')
print ('The size of our training "y" (output feature) is',
y_train.shape)
print ('\n')
```

```
print ('The size of our testing "y" (output features) is',
y_test.shape)

from sklearn.svm import SVC
svc_model = SVC()

print(svc_model.fit(X_train, y_train))
y_predict = svc_model.predict(X_test)
score = svc_model.score(X_test, y_test)
print("Test Accuracy: ", score)
score = svc_model.score(X_train, y_train)
print("Train Accuracy: ", score)
```




diagnosis
2
4

5.2 *Preprocessing Data*

Pada tahap ini, penulis akan melakukan *outlier handling* saja karena tidak ada *missing value* pada data yang digunakan. Proses untuk melakukan *preprocessing data* yaitu dengan menggunakan bahasa pemrograman Python, dimana *source code* tersaji sebagai berikut.

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('data.csv', header=0)

#PREPROCESSING DATA
# dataset.replace('?', -99999, inplace=True) #-9999 biar outlier,
gak masuk ke grafik
dataset.drop("id",1)
mapping={'M':4, 'B':2}
print(dataset.shape)
dataset['diagnosis'] = dataset['diagnosis'].map(mapping)
X = dataset.iloc[:, 1:31].values # parameter yang mau di train
y = dataset.iloc[:, 1].values # target

# Membagi data set menjadi 80% data training dan 20% data testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Menggunakan Kernel PCA untuk Visualisasi data
from sklearn.decomposition import KernelPCA
kpca = KernelPCA(n_components = 2, kernel = 'rbf')
X_train = kpca.fit_transform(X_train)
X_test = kpca.transform(X_test)
```

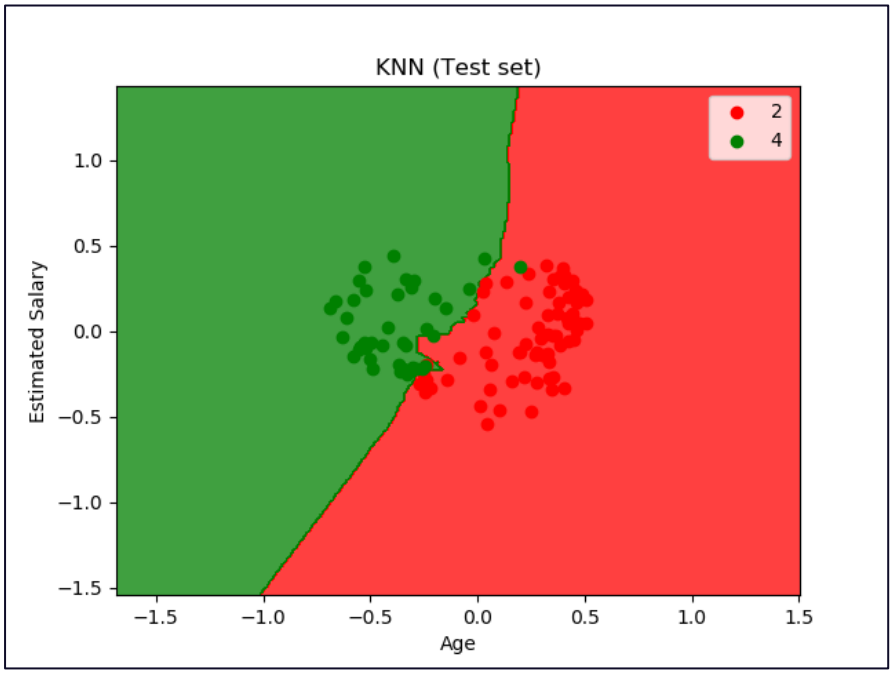
5.3 Klasifikasi

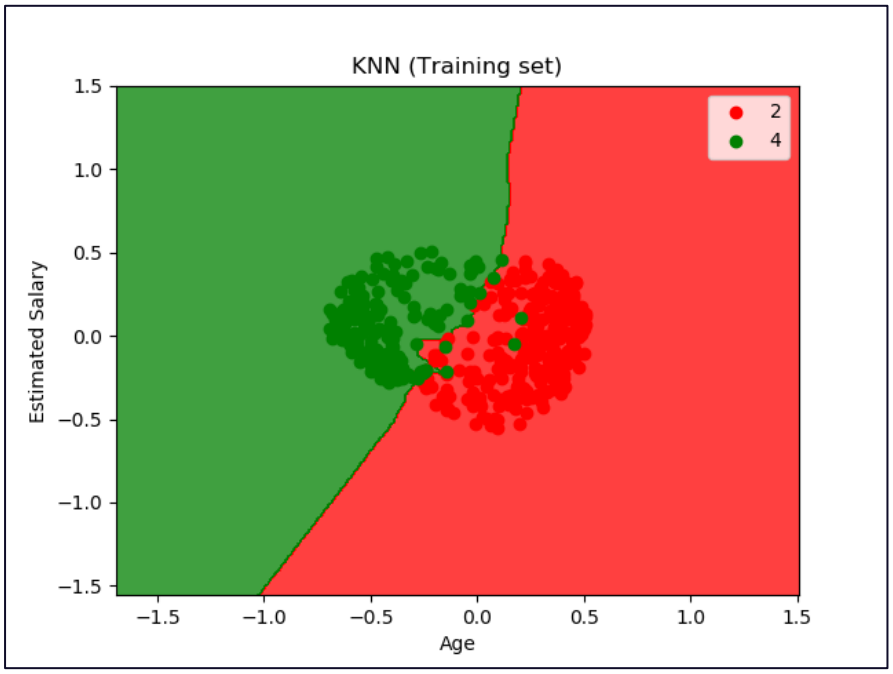
Kemudian, setelah dilakukan *preprocessing* data, dilanjutkan dengan proses klasifikasi dengan menggunakan algoritma kNN, SVM, dan Naïve Bayes pada Python sebagai berikut ini.

```
# Fitting K-NN to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
labels = ['y_test', 'y_pred']
cm = confusion_matrix(y_test, y_pred)
print(cm)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

# Checking Accuracy
accuracy = classifier.score(X_test, y_test)
print("Test Accuracy: ", accuracy)
accuracy = classifier.score(X_train, y_train)
print("Train Accuracy: ", accuracy)

# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
```

```
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('KNN (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()

# Visualising the Test set results
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('KNN (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

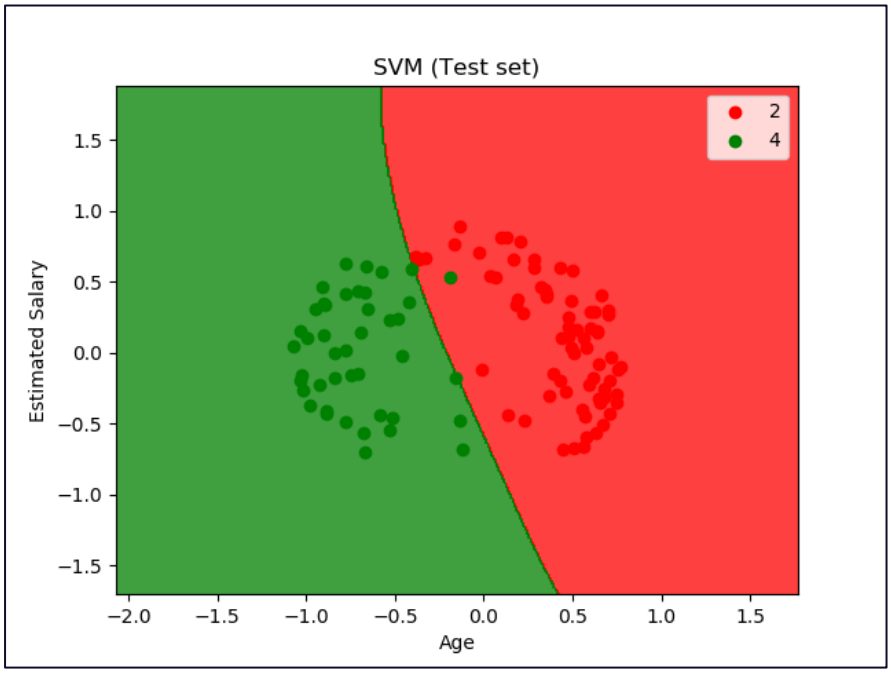
Berikut ini adalah hasil klasifikasi dari *dataset* kanker payudara dengan menggunakan KNN, SVM, dan Naïve Bayes.





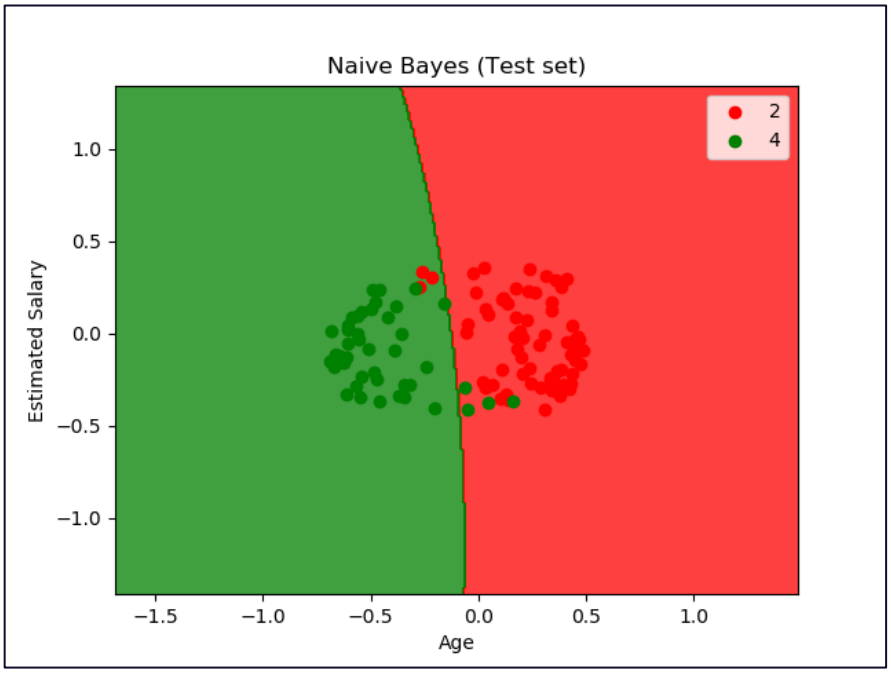
Test Accuracy: $0.969230769230769 \approx 96.92\%$

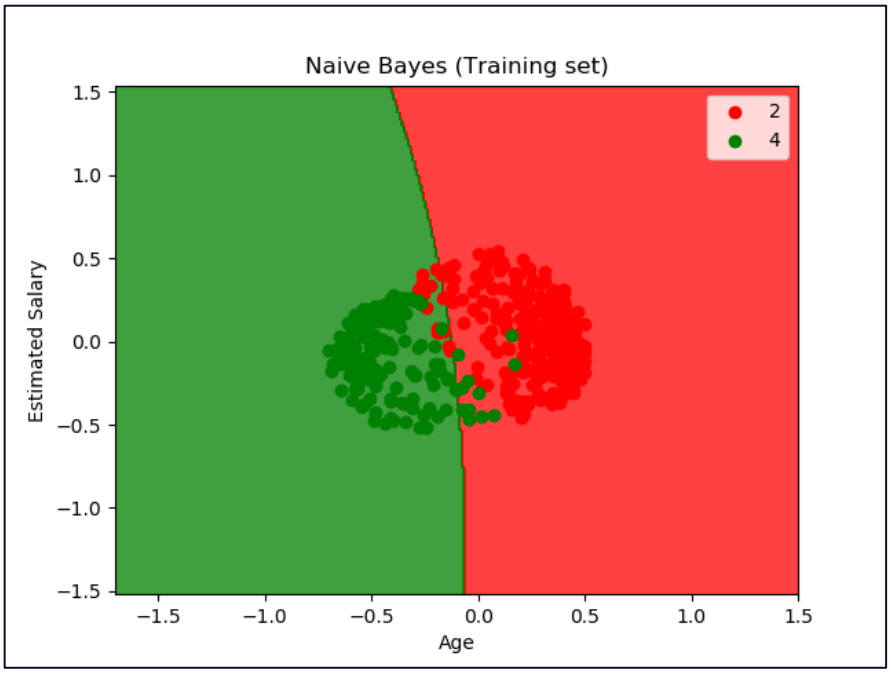
Train Accuracy: $0.9736842105263158 \approx 97.37\%$



Test Accuracy: $0.9473684210526315 \approx 94.74\%$

Train Accuracy: $0.9736263736263732 \approx 97.36\%$





Test Accuracy: $0.91228070175432 \approx 91.22\%$

Train Accuracy: $0.94065934065934 \approx 94.06\%$

5.4 *Clustering*

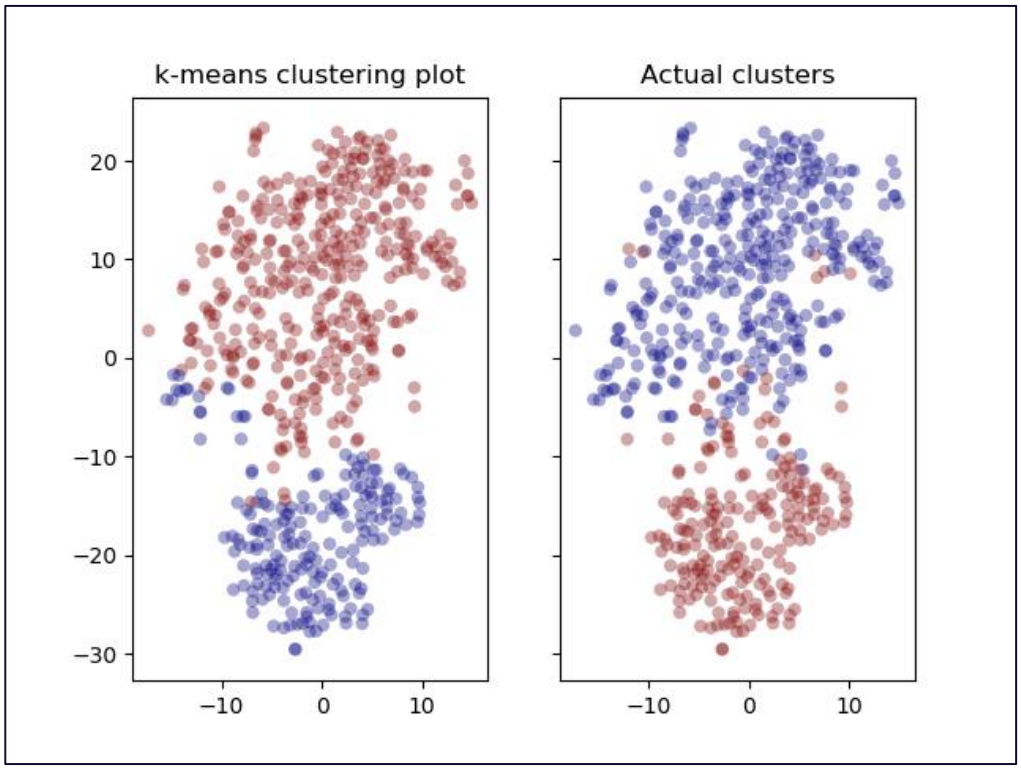
Berikut adalah *source code* proses *clustering* menggunakan algoritma *k-means*, *Spectral Clustering*, dan Hierarchical Clustering yang diimplementasikan dengan menggunakan Python.

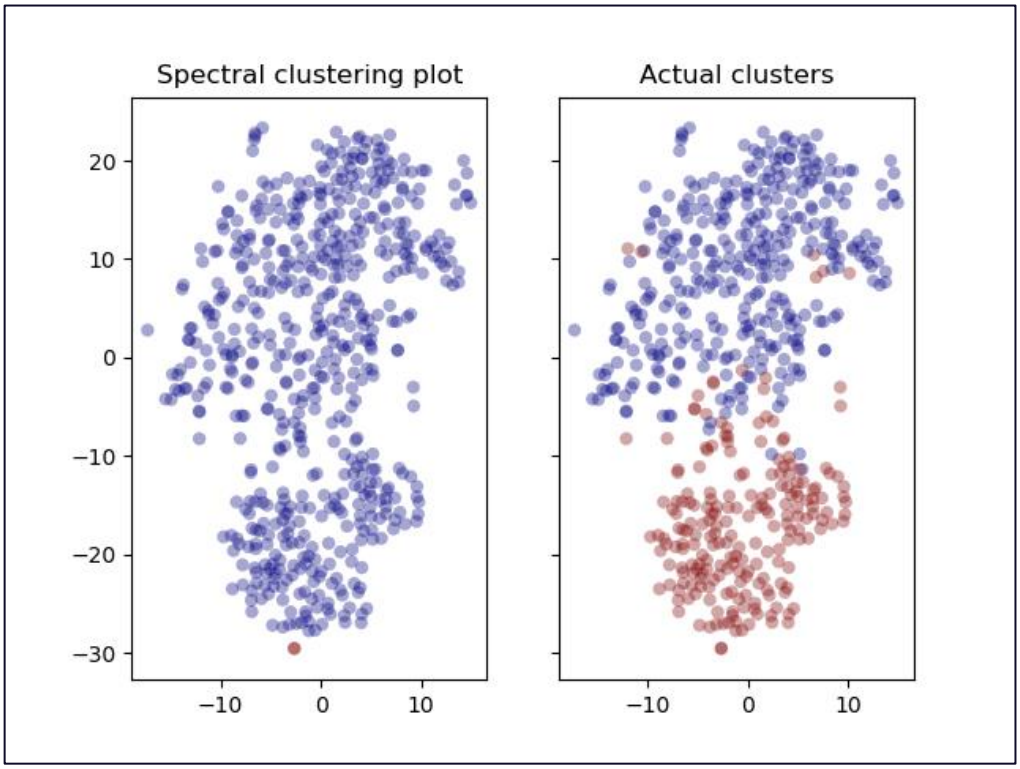
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g.
pd.read_csv)

import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt2
import matplotlib.cm as cm
from sklearn import preprocessing
from subprocess import check_output

# dataset
data = pd.read_csv('../classification/data.csv')

# Cleaning and modifying the data
data = data.drop('id',axis=1)
data = data.drop('Unnamed: 32',axis=1)
# Mapping Benign to 0 and Malignant to 1
data['diagnosis'] = data['diagnosis'].map({'M':1,'B':0})
# Scaling the dataset
datas = pd.DataFrame(preprocessing.scale(data.iloc[:,1:32]))
datas.columns = list(data.iloc[:,1:32].columns)
datas['diagnosis'] = data['diagnosis']
# Creating the high dimensional feature space X
data_drop = datas.drop('diagnosis',axis=1)
X = data_drop.values


#Creating a 2D visualization to visualize the clusters
from sklearn.manifold import TSNE
tsne = TSNE(verbose=1, perplexity=40, n_iter= 10000)
Y = tsne.fit_transform(X)

#Cluster using k-means
from sklearn.cluster import KMeans
kmns = KMeans(n_clusters=2, init='k-means++', n_init=10,
max_iter=3000, tol=0.0001, precompute_distances='auto', verbose=1,
random_state=None, copy_x=True, n_jobs=1, algorithm='auto')
```

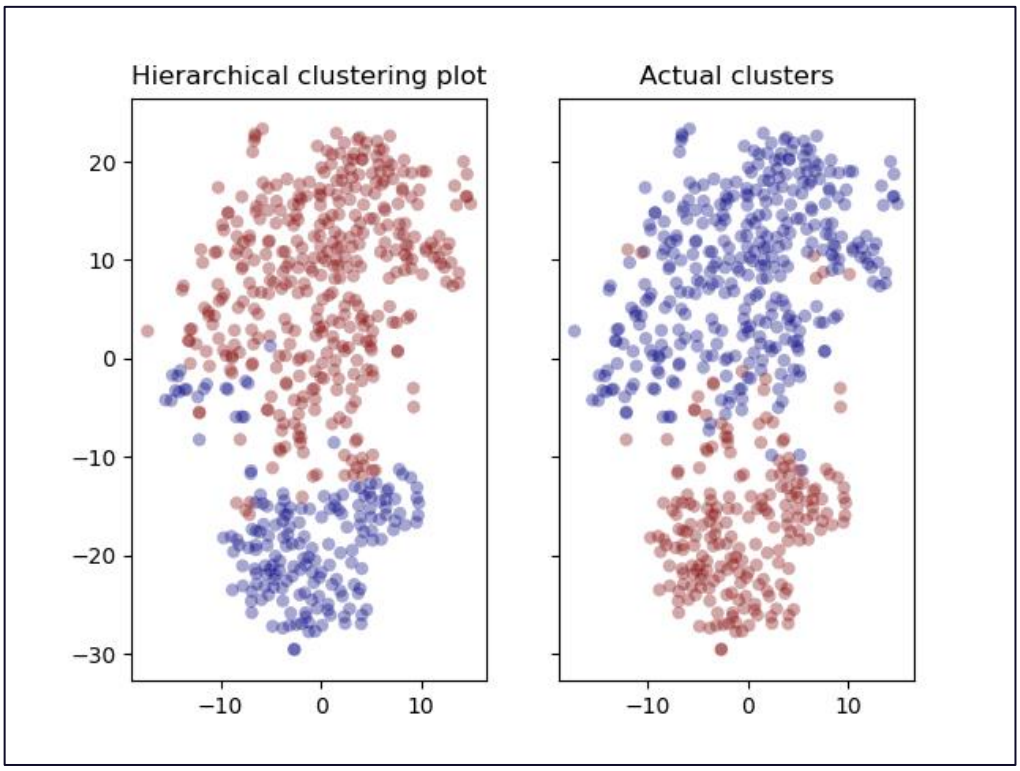
```
kY = kmns.fit_predict(X)

f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)

ax1.scatter(Y[:,0],Y[:,1],  c=kY, cmap = "jet", edgecolor = "None",
alpha=0.35)
ax1.set_title('k-means clustering plot')

ax2.scatter(Y[:,0],Y[:,1],  c = datas['diagnosis'], cmap = "jet",
edgecolor = "None", alpha=0.35)
ax2.set_title('Actual clusters')
plt2.show()

#Cluster using Spectral clustering and visualize using Isomap
from sklearn.cluster import SpectralClustering


# Play with gamma to optimize the clustering results
kmns = SpectralClustering(n_clusters=2,  gamma=0.5, affinity='rbf',
eigen_tol=0.0, assign_labels='kmeans', degree=3, coef0=1,
kernel_params=None, n_jobs=1)
kY = kmns.fit_predict(X)

f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)


ax1.scatter(Y[:,0],Y[:,1],  c=kY, cmap = "jet", edgecolor = "None",
alpha=0.35)
ax1.set_title('Spectral clustering plot')

ax2.scatter(Y[:,0],Y[:,1],  c = datas['diagnosis'], cmap = "jet",
edgecolor = "None", alpha=0.35)
ax2.set_title('Actual clusters')
plt2.show()

# Cluster using hierarchical clustering
from sklearn.cluster import AgglomerativeClustering
aggC = AgglomerativeClustering(n_clusters=2, linkage='ward')
kY = aggC.fit_predict(X)


f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)
```

```
ax1.scatter(Y[:,0],Y[:,1],  c=kY, cmap = "jet", edgecolor = "None",
alpha=0.35)
ax1.set_title('Hierarchical clustering plot')

ax2.scatter(Y[:,0],Y[:,1],  c = datas['diagnosis'], cmap = "jet",
edgecolor = "None", alpha=0.35)
ax2.set_title('Actual clusters')
plt2.show()
#dendogram
from scipy.cluster.hierarchy import dendrogram, linkage
Z = linkage(X)
dendrogram(Z)
plt.show()
```

Hasil dari cuplikan *source code* diatas adalah sebagai berikut ini.

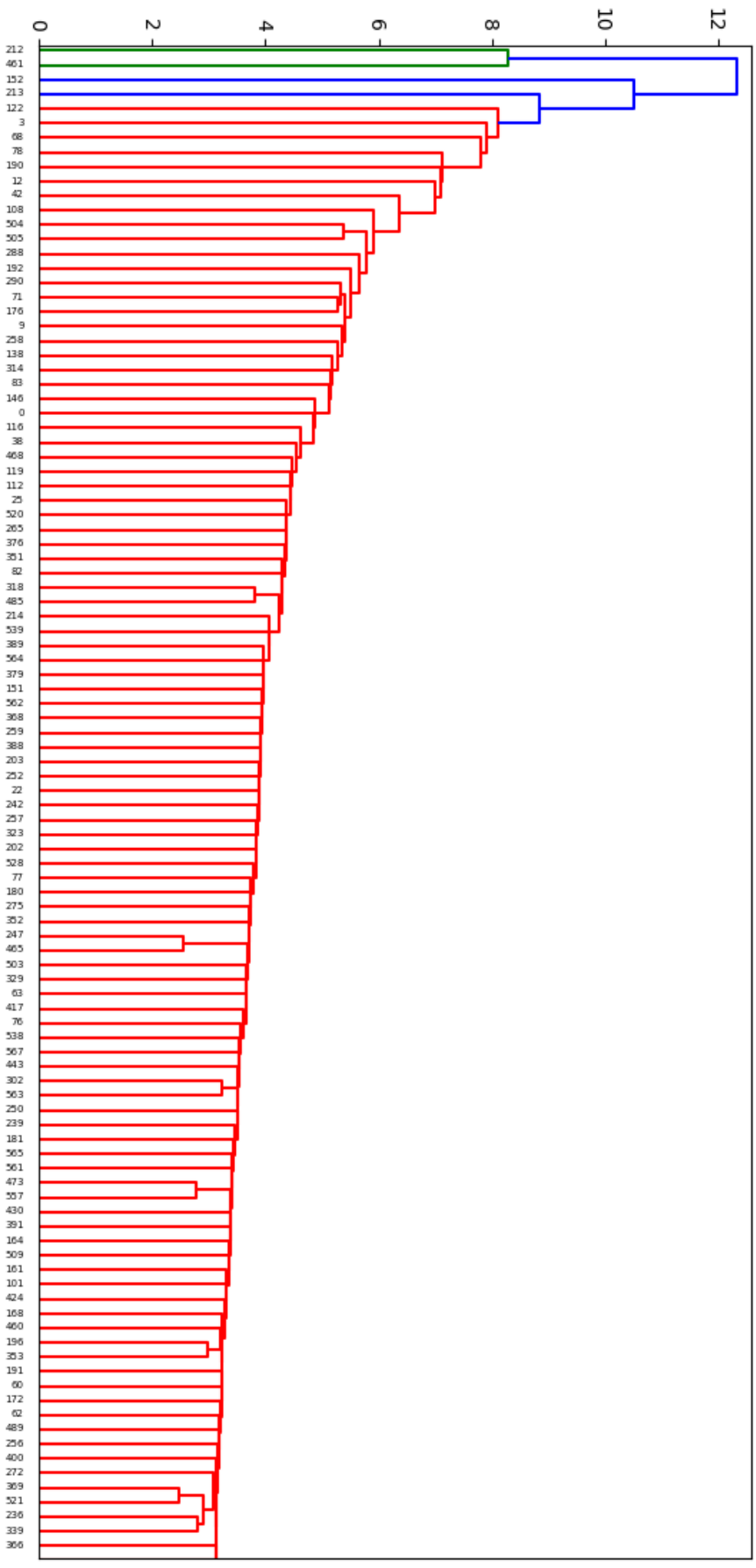


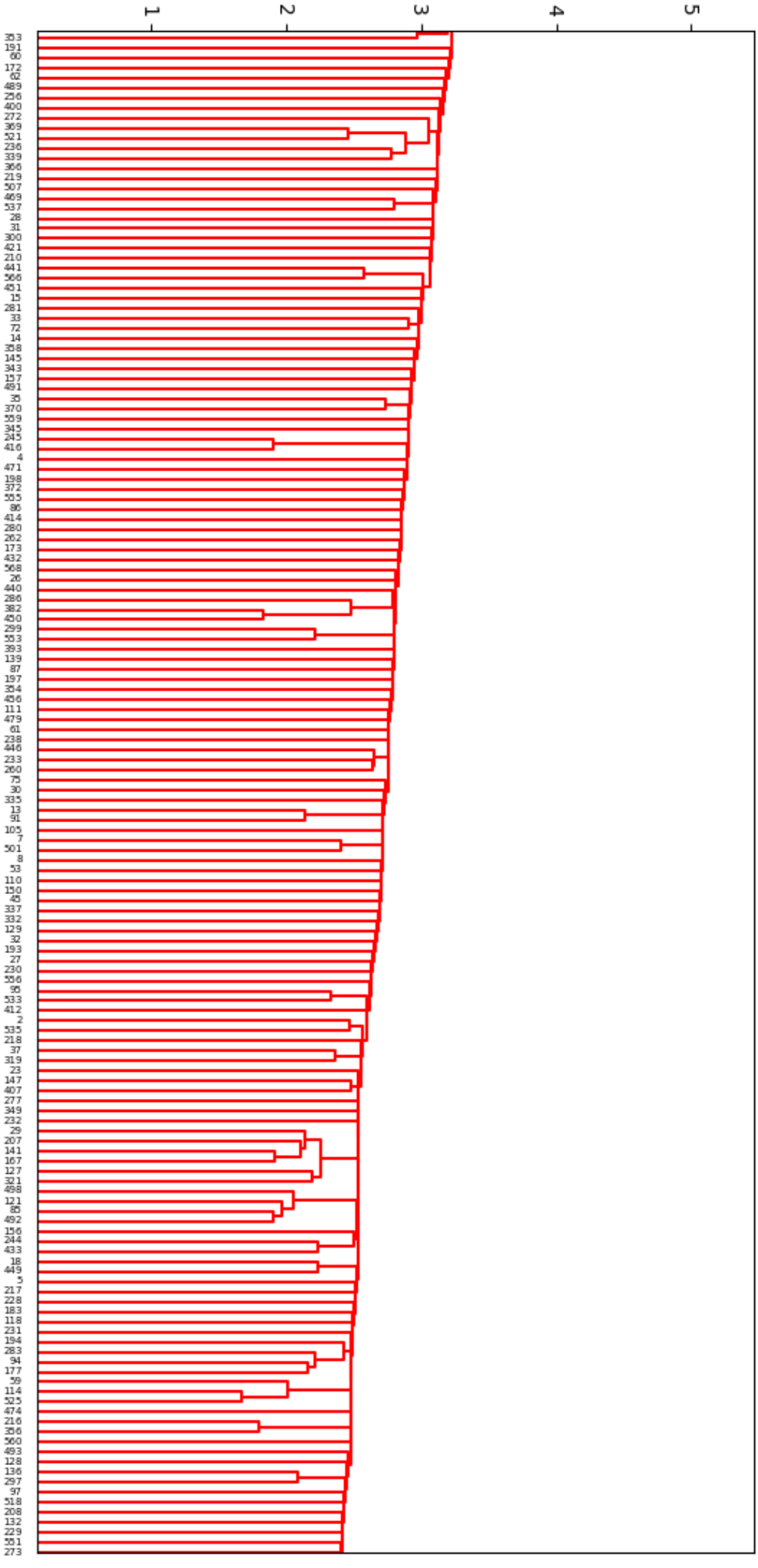

Spectral clustering plot
Actual clusters
20
10
0
−10
−20
−30
−10
0
10
−10
0
10

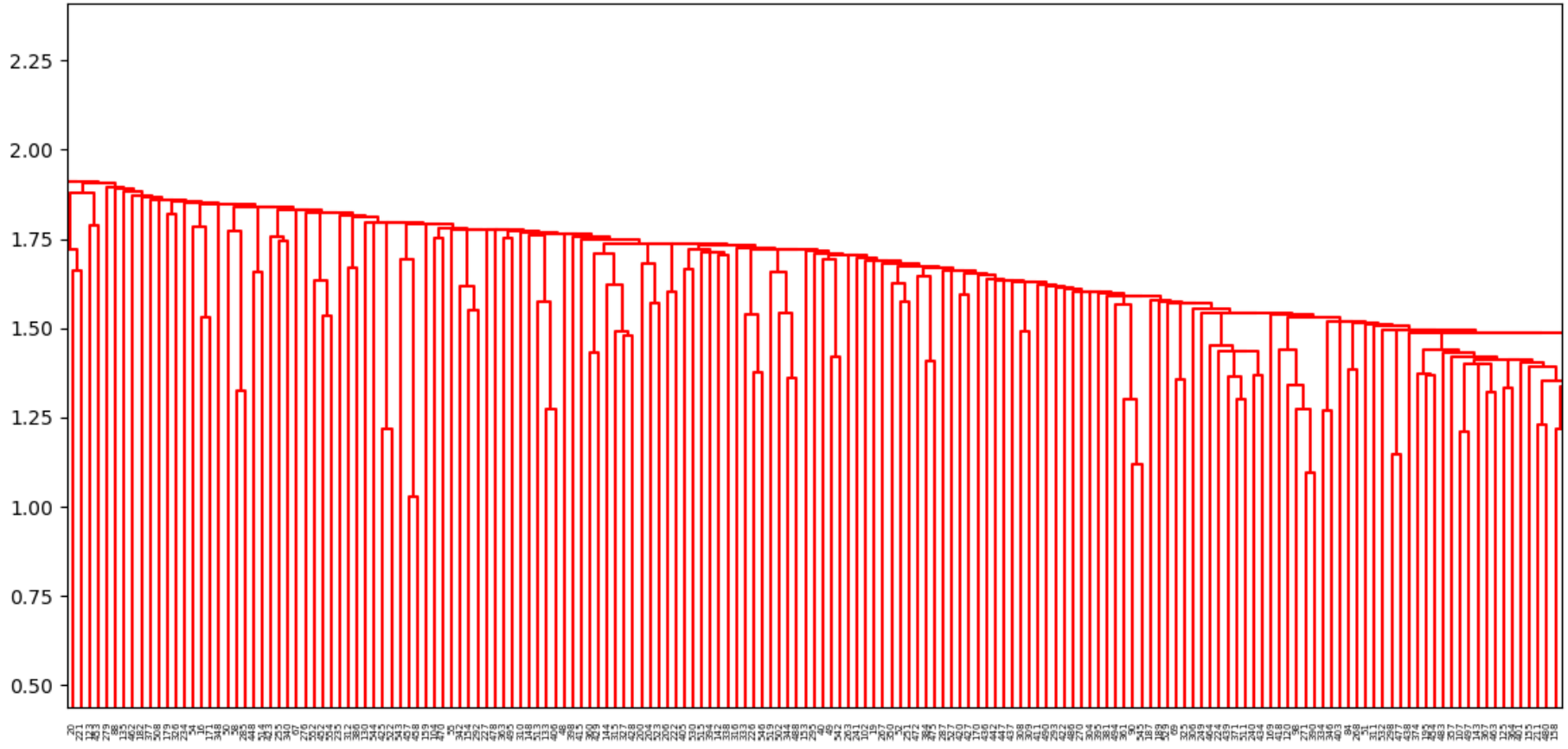

Hierarchical clustering plot
Actual clusters
20
10
0
−10
−20
−30
−10
0
10
−10
0
10

Dendogram dari *hierarchical clustering*

Hasil perbesaran




1
2
3
4
5
353
191
60
172
62
489
256
400
272
369
521
236
339
306
219
507
469
537
28
31
300
421
210
441
566
451
15
281
33
72
14
358
145
343
157
491
35
370
559
345
245
416
4
471
198
372
555
86
414
280
262
173
432
568
26
440
286
382
450
299
553
393
139
87
197
354
456
111
479
61
238
446
233
260
75
30
335
13
91
105
7
501
8
53
110
150
45
337
332
129
32
193
27
230
556
95
533
412
2
535
218
37
319
23
147
407
277
349
232
29
207
141
167
127
321
498
121
85
492
156
244
433
18
449
5
217
228
183
118
231
194
283
94
177
59
114
525
474
216
356
560
493
128
136
297
97
518
208
132
229
551
273


2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50

5.5 ***Performance Analysis*** **(***Precision, Recall, f1 score***)**

Kemudian, berdasarkan hasil visualisasi klasifikasi diatas, penulis juga menyajikan visualisasi tabel *confusion matrix* yang diperoleh sebagai berikut.

kNN

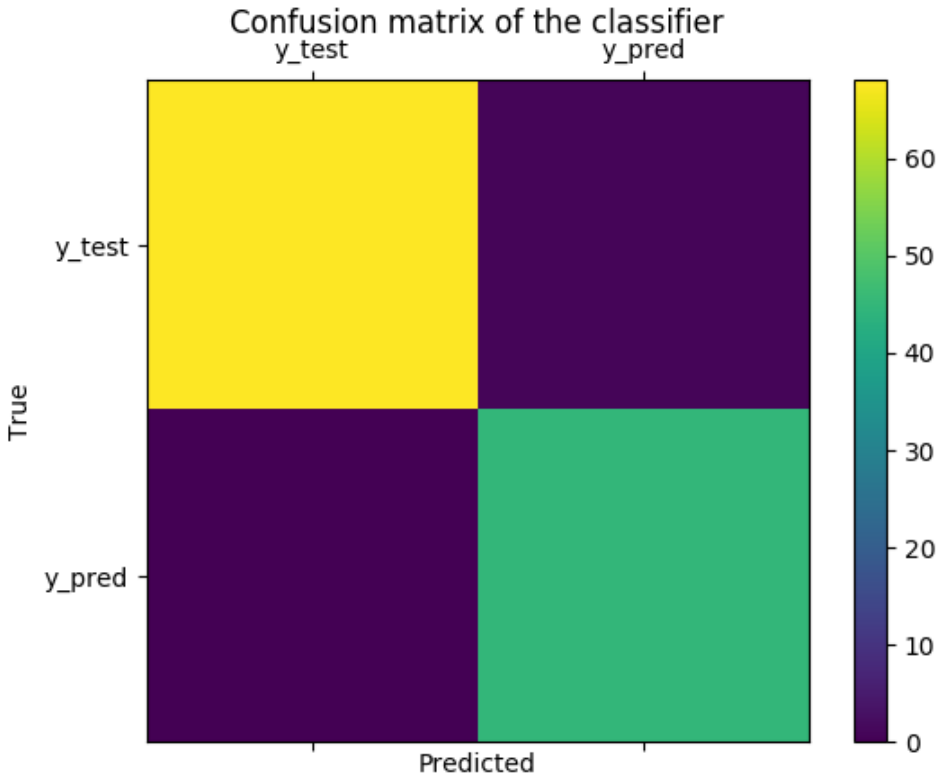
Type	Precision	recall	f1-score	support
2	1.00	0.96	0.98	70
4	0.94	1.00	0.97	44
micro avg	0.97	0.97	0.97	114
macro avg	0.97	0.98	0.97	114
weighted avg	0.98	0.97	0.97	114

SVM

Type	Precision	recall	f1-score	support
2	1.00	0.93	0.96	70
4	0.90	1.00	0.95	44
micro avg	0.96	0.96	0.97	114
macro avg	0.97	0.96	0.95	114
weighted avg	0.96	0.96	0.96	114

NaiveBayes

Type	Precision	recall	f1-score	support
2	0.98	0.91	0.95	70
4	0.88	0.98	0.92	44
micro avg	0.94	0.94	0.94	114
macro avg	0.93	0.95	0.94	114
weighted avg	0.94	0.94	0.94	114



5.6 Prediksi dengan *Multi-Layer Perceptron*

Berikut ini akan dilakukan prediksi hasil dengan *data testing* yang diacak (*shuffle*) dengan menggunakan **`sklearn.utils.shuffle`** yang merupakan *library* Python.

```
import tensorflow as tf
import pandas as pd
from sklearn.utils import shuffle
import matplotlib.gridspec as gridspec
import seaborn as sns
import matplotlib.pyplot as plt


# Define methods for printing data into graphs
def GraphAreaMeanFeature():
    # Compare the mean area on benign vs malignant tumors
    print("Malignant")
    print(train_data.area_mean[train_data.diagnosis ==
"M"].describe())
    print()
    print("Benign")
    print(train_data.area_mean[train_data.diagnosis ==
"B"].describe())

    f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(12,
4))

    bins = 50

    ax1.hist(train_data.area_mean[train_data.diagnosis == "M"],
bins=bins)
    ax1.set_title('Malignant')

    ax2.hist(train_data.area_mean[train_data.diagnosis == "B"],
bins=bins)
    ax2.set_title('Benign')

    plt.xlabel('Area Mean')
    plt.ylabel('Number of Diagnosis')
    plt.savefig('graphs/MalignantVsBenign-MeanAreaPlot.png')
    plt.show()
```

```
def GraphAreaWorstFeature():
    print("Malignant")
    print(train_data.area_worst[train_data.diagnosis ==
"M"].describe())
    print()
    print("Benign")
    print(train_data.area_worst[train_data.diagnosis ==
"B"].describe())

    f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(12,
4))

    bins = 30

    ax1.hist(train_data.area_worst[train_data.diagnosis == "M"],
bins=bins)
    ax1.set_title('Malignant')

    ax2.hist(train_data.area_worst[train_data.diagnosis == "B"],
bins=bins)
    ax2.set_title('Benign')

    plt.xlabel('Area Worst')
    plt.ylabel('Number of Diagnosis')
    plt.yscale('log')
    plt.savefig('graphs/MalignantVsBenign-WorstAreaPlot.png')
    plt.show()


def GraphRestOfFeatures():
    # Select only the rest of the features.
    r_data = train_data.drop([idKey, areaMeanKey, areaWorstKey,
diagnosisKey], axis=1)
    r_features = r_data.columns

    plt.figure(figsize=(12, 28 * 4))
    gs = gridspec.GridSpec(28, 1)
    for i, cn in enumerate(r_data[r_features]):
        ax = plt.subplot(gs[i])
        sns.distplot(train_data[cn][train_data.diagnosis == "M"],
bins=50)
        sns.distplot(train_data[cn][train_data.diagnosis == "B"],
bins=50)
        ax.set_xlabel('')
        ax.set_title('histogram of feature: ' + str(cn))
```

```
    plt.savefig('graphs/MalignantVsBenign-
RestOfFeaturesHistogram.png')
    plt.show()


# Get the training data file
train_filename = "../classification/data.csv"

# Define column keys for data.csv
idKey = "id"
diagnosisKey = "diagnosis"
radiusMeanKey = "radius_mean"
textureMeanKey = "texture_mean"
perimeterMeanKey = "perimeter_mean"
areaMeanKey = "area_mean"
smoothnessMeanKey = "smoothness_mean"
compactnessMeanKey = "compactness_mean"
concavityMeanKey = "concavity_mean"
concavePointsMeanKey = "concave points_mean"
symmetryMeanKey = "symmetry_mean"
fractalDimensionMean = "fractal_dimension_mean"
radiusSeKey = "radius_se"
textureSeKey = "texture_se"
perimeterSeKey = "perimeter_se"
areaSeKey = "area_se"
smoothnessSeKey = "smoothness_se"
compactnessSeKey = "compactness_se"
concavitySeKey = "concavity_se"
concavePointsSeKey = "concave points_se"
symmetrySeKey = "symmetry_se"
fractalDimensionSeKey = "fractal_dimension_se"
radiusWorstKey = "radius_worst"
textureWorstKey = "texture_worst"
perimeterWorstKey = "perimeter_worst"
areaWorstKey = "area_worst"
smoothnessWorstKey = "smoothness_worst"
compactnessWorstKey = "compactness_worst"
concavityWorstKey = "concavity_worst"
concavePointsWorstKey = "concave points_worst"
symmetryWorstKey = "symmetry_worst"
fractalDimensionWorstKey = "fractal_dimension_worst"

# Columns used for training
```

```
train_columns = [idKey, diagnosisKey, radiusMeanKey,
textureMeanKey, perimeterMeanKey, areaMeanKey, smoothnessMeanKey,
                 compactnessMeanKey, concavityMeanKey,
concavePointsMeanKey, symmetryMeanKey, fractalDimensionMean,
                 radiusSeKey, textureSeKey, perimeterSeKey,
areaSeKey, smoothnessSeKey, compactnessSeKey,
                 concavitySeKey, concavePointsSeKey, symmetrySeKey,
fractalDimensionSeKey, radiusWorstKey,
                 textureWorstKey, perimeterWorstKey, areaWorstKey,
smoothnessWorstKey, compactnessWorstKey,
                 concavityWorstKey, concavePointsWorstKey,
symmetryWorstKey, fractalDimensionWorstKey]


# Method for loading the training data
def GetTrainingData():
    dataFile = pd.read_csv(train_filename, names=train_columns,
delimiter=',', skiprows=1)
    return dataFile


# Load the training data
train_data = GetTrainingData()

# Plot the data graphs
# GraphAreaMeanFeature()
# GraphAreaWorstFeature()
# GraphRestOfFeatures()

train_data.head()
train_data.describe()
train_data.isnull().sum()

# Update the value of diagnosis (1 = malignant and 0 = benign)
train_data.loc[train_data.diagnosis == 'M', 'diagnosis'] = 1
train_data.loc[train_data.diagnosis == 'B', 'diagnosis'] = 0

# Create a new feature for benign (non-malignant) diagnosis
train_data.loc[train_data.diagnosis == 0, 'benign'] = 1
train_data.loc[train_data.diagnosis == 1, 'benign'] = 0

# Convert benign column type to integer
train_data['benign'] = train_data.benign.astype(int)

# Rename 'Class' to 'Malignant'
```

```
train_data = train_data.rename(columns={'diagnosis': 'malignant'})

pd.set_option("display.max_columns", 101)
train_data.head()

# Print result stats
# print(train_data.benign.value_counts())
# print()
# print(train_data.malignant.value_counts())


# Create dataframes for only Malignant and Benign diagnosis
Malignant = train_data[train_data.malignant == 1]
Benign = train_data[train_data.benign == 1]

# Set train_X = 80% of the malignant diagnosis
train_X = Malignant.sample(frac=0.8)
count_Malignants = len(train_X)

# Add 805 of benign diagnosis to the train_X set
train_X = pd.concat([train_X, Benign.sample(frac=0.8)], axis=0)

# Text_X dataset should contain all the diagnostics not present in
train_X
test_X = train_data.loc[~train_data.index.isin(train_X.index)]

# Shuffle the data frames for training to be done in random order
train_X = shuffle(train_X)
test_X = shuffle(test_X)

# Add target features to train_Y and test_Y
train_Y = train_X.malignant
train_Y = pd.concat([train_Y, train_X.benign], axis=1)

test_Y = test_X.malignant
test_Y = pd.concat([test_Y, test_X.benign], axis=1)

# Drop target features from train_X and test_X
train_X = train_X.drop(['malignant', 'benign'], axis=1)
test_X = test_X.drop(['malignant', 'benign'], axis=1)

# Check if all training/testing dataframes are of the right length
# print(len(train_X))
# print(len(train_Y))
# print(len(test_X))
```

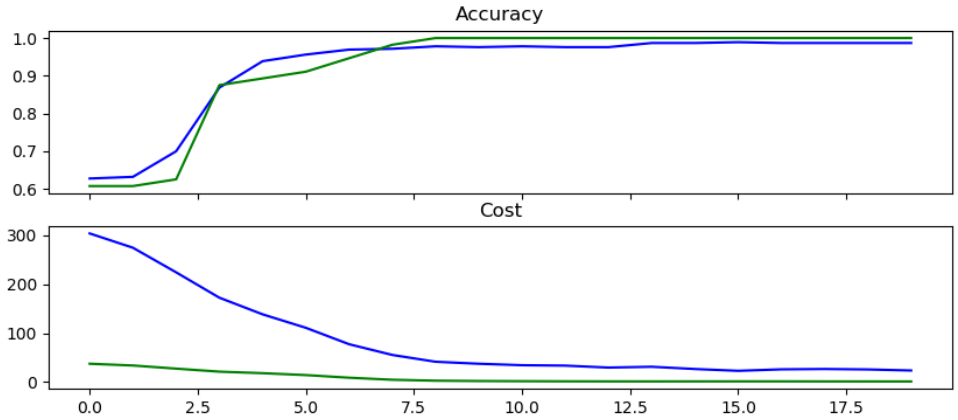
```
# print(len(test_Y))


# Names of all the features in train_X
features = train_X.columns.values

# Transform each feature in features so that it has a mean of 0 and
s.d. of 1. (Helps training the softmax algorithm)
for feature in features:
    mean, std = train_data[feature].mean(),
train_data[feature].std()
    train_X.loc[:, feature] = (train_X[feature] - mean) / std
    test_X.loc[:, feature] = (test_X[feature] - mean) / std

# Training the Neural Network

# Neural Network Parameters
learning_rate = 0.005
training_dropout = 0.9
display_step = 1
batch_size = 100
accuracy_history = []
cost_history = []
valid_accuracy_history = []
valid_cost_history = []

# Number of input nodes
input_nodes = train_X.shape[1]

# Number of labels (malignant and benign)
num_labels = 2

# Split the testing data into validation and testing sets
split = int(len(test_Y) / 2)

train_size = train_X.shape[0]
n_samples = train_Y.shape[0]

input_X = train_X.values
input_Y = train_Y.values
input_X_valid = test_X.values[:split]
input_Y_valid = test_Y.values[:split]
input_X_test = test_X.values[split:]
input_Y_test = test_Y.values[split:]
```

```
def CalculateHiddenNodes(nodes):
    return (((2 * nodes) / 3) + num_labels)


# Number of nodes in each hidden layer
hidden_nodes1 = round(CalculateHiddenNodes(input_nodes))
hidden_nodes2 = round(CalculateHiddenNodes(hidden_nodes1))
hidden_nodes3 = round(CalculateHiddenNodes(hidden_nodes2))

print(input_nodes, hidden_nodes1, hidden_nodes2, hidden_nodes3)

# Percent of nodes to keep during dropout
pkeep = tf.placeholder(tf.float32)

# Input
x = tf.placeholder(tf.float32, [None, input_nodes])

# Layer 1
W1 = tf.Variable(tf.truncated_normal([input_nodes, hidden_nodes1],
stddev=0.1))
b1 = tf.Variable(tf.zeros([hidden_nodes1]))
y1 = tf.nn.relu(tf.matmul(x, W1) + b1)

# Layer 2
W2 = tf.Variable(tf.truncated_normal([hidden_nodes1,
hidden_nodes2], stddev=0.1))
b2 = tf.Variable(tf.zeros([hidden_nodes2]))
y2 = tf.nn.relu(tf.matmul(y1, W2) + b2)

# Layer 3
W3 = tf.Variable(tf.truncated_normal([hidden_nodes2,
hidden_nodes3], stddev=0.1))
b3 = tf.Variable(tf.zeros([hidden_nodes3]))
y3 = tf.nn.relu(tf.matmul(y2, W3) + b3)
y3 = tf.nn.dropout(y3, pkeep)

# Layer 4
W4 = tf.Variable(tf.truncated_normal([hidden_nodes3, 2],
stddev=0.1))
b4 = tf.Variable(tf.zeros([2]))
y4 = tf.nn.softmax(tf.matmul(y3, W4) + b4)

# Output
y = y4
y_ = tf.placeholder(tf.float32, [None, num_labels])
```

```
# Minimise error using cross entropy
cost = -tf.reduce_sum(y_ * tf.log(y))

# Adam Optimiser
optimiser = tf.train.AdamOptimizer(learning_rate).minimize(cost)

# Test Model
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))

# Calculate accuracy
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

# Initialise variables
init = tf.global_variables_initializer()

# Launch the graph
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    for epoch in range(20):
        for batch in range(int(n_samples / batch_size)):
            batch_x = input_X[batch * batch_size: (1 + batch) *
batch_size]
            batch_y = input_Y[batch * batch_size: (1 + batch) *
batch_size]

            sess.run([optimiser], feed_dict={
                x: batch_x,
                y_: batch_y,
                pkeep: training_dropout
            })

        train_accuracy, newCost = sess.run([accuracy, cost],
                                           feed_dict={x: input_X,
y_: input_Y,
                                                      pkeep:
training_dropout})

        valid_accuracy, valid_newCost = sess.run([accuracy, cost],
                                                 feed_dict={x:
input_X_valid,
                                                            y_:
input_Y_valid, pkeep: 1})
```

```
        print("Epoch: ", epoch, "  Accuracy: ",
"{:.5f}".format(train_accuracy),
                "  Cost: ", "{:.5f}".format(newCost),
                "  Valid Accuracy: ",
"{:.5f}".format(valid_accuracy),
                "  Valid Cost: ", "{:.5f}".format(valid_newCost))

            # Record the results of the results of the model
        accuracy_history.append(train_accuracy)
        cost_history.append(newCost)
        valid_accuracy_history.append(valid_accuracy)
        valid_cost_history.append(valid_newCost)

        # If the model does n ot improve after 15 logs, stop the
training
        if valid_accuracy < max(valid_accuracy_history) and epoch >
100:
            stop_early += 1
            if stop_early == 15:
                break
        else:
            stop_early = 0

    print("Run Complete Finished.")

    # Plot the accuracy and cost summaries
    f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(10,
4))

    ax1.plot(accuracy_history, color='b')  # blue
    ax1.plot(valid_accuracy_history, color='g')  # green
    ax1.set_title('Accuracy')

    ax2.plot(cost_history, color='b')
    ax2.plot(valid_cost_history, color='g')
    ax2.set_title('Cost')

    plt.xlabel('Epochs (x10)')
    plt.savefig('graphs/FinalAccuracyAndCostSummary.png')
    plt.show()
```

Berdasarkan hasil *source code* tersebut, kita memperoleh hasil prediksi dengan *Multi-Layer Perceptron* sebagai berikut.




```
31 23 17 13 (input_nodes: train_X.shape, hidden_nodes1:
input_nodes, hidden_nodes2, hidden_nodes3)

Epoch: 0 Accuracy: 0.62719 Cost: 303.86292 Valid Accuracy: 0.60714
Valid Cost: 37.30396

Epoch: 1 Accuracy: 0.63158 Cost: 274.62439 Valid Accuracy: 0.60714
Valid Cost: 33.73276

Epoch: 2 Accuracy: 0.69956 Cost: 224.27368 Valid Accuracy: 0.62500
Valid Cost: 27.30453

Epoch: 3 Accuracy: 0.86842 Cost: 172.38452 Valid Accuracy: 0.87500
Valid Cost: 21.12439

Epoch: 4 Accuracy: 0.93860 Cost: 138.36185 Valid Accuracy: 0.89286
Valid Cost: 17.99998

Epoch: 5 Accuracy: 0.95614 Cost: 110.83379 Valid Accuracy: 0.91071
Valid Cost: 14.10146

Epoch: 6 Accuracy: 0.96930 Cost: 77.26272 Valid Accuracy: 0.94643
Valid Cost: 8.75187

Epoch: 7 Accuracy: 0.97149 Cost: 55.30008 Valid Accuracy: 0.98214
Valid Cost: 4.56376
```

```
Epoch: 8 Accuracy: 0.97807 Cost: 41.37629 Valid Accuracy: 1.00000
Valid Cost: 2.64346

Epoch: 9 Accuracy: 0.97588 Cost: 37.30238 Valid Accuracy: 1.00000
Valid Cost: 1.97846

Epoch: 10 Accuracy: 0.97807 Cost: 34.33464 Valid Accuracy: 1.00000
Valid Cost: 1.60540

Epoch: 11 Accuracy: 0.97588 Cost: 33.48156 Valid Accuracy: 1.00000
Valid Cost: 1.33480

Epoch: 12 Accuracy: 0.97588 Cost: 29.47034 Valid Accuracy: 1.00000
Valid Cost: 1.19615

Epoch: 13 Accuracy: 0.98684 Cost: 31.13475 Valid Accuracy: 1.00000
Valid Cost: 1.16577

Epoch: 14 Accuracy: 0.98684 Cost: 26.50785 Valid Accuracy: 1.00000
Valid Cost: 1.20009

Epoch: 15 Accuracy: 0.98904 Cost: 22.84942 Valid Accuracy: 1.00000
Valid Cost: 1.25582

Epoch: 16 Accuracy: 0.98684 Cost: 25.87312 Valid Accuracy: 1.00000
Valid Cost: 1.25636

Epoch: 17 Accuracy: 0.98684 Cost: 26.54008 Valid Accuracy: 1.00000
Valid Cost: 1.16558

Epoch: 18 Accuracy: 0.98684 Cost: 25.67315 Valid Accuracy: 1.00000
Valid Cost: 1.10405

Epoch: 19 Accuracy: 0.98684 Cost: 23.48993 Valid Accuracy: 1.00000
Valid Cost: 1.06180

Run Complete Finished
```

DAFTAR PUSTAKA

[1] Mukhlash, Imam. 2018. *Catatan Kuliah Data Mining Departemen Matematika FMKSD ITS.* Surabaya : Institut Teknologi Sepuluh Nopember

[2] Han, Jiawei, dkk. 2012. *Data Mining : Concepts and Techniques.* United States of America : Elsevier

[3] Soelaiman, Rully. 2006. *Analisis Kinerja Algoritma Prefixspan dan Aprioriall Pada Penggalian Pola Sekuensial.* Yogyakarta : SNATI

[4] Abbas, Irfan. 2016. *Penerapan Algoritma Support Vector Machine (SVM) untuk Memprediksi dan Membandingkan Hasil Trend Kurva pada Trading Forex*. Gorontalo : STMIK Ichsan

[5] Wibowo, Madha C., dkk. 2013. Pengenalan Pola Tulisan Tangan Aksara Jawa HA NA CA RA KA menggunakan Multi-Layer Perceptron. Surabaya : STIKOM

[6] *https://medium.com/@infharis/data-mining-definisi-dan-cara-kerja-algoritma-apriori-untuk-pencarian-association-rule-a44a8f864a61* diakses pada 06 Desember 2018

[7] *https://medium.com/python-pandemonium/introduction-to-exploratory-data-analysis-in-python-8b6bcb55c190* diakses pada 07 Desember 2018

[8] *http://www.bigdatasharingvision.com/articles/support-vector-machine* diakses pada 08 Desember 2018

[9] *https://www.advernesia.com/blog/data-science/pengertian-dan-cara-kerja-algoritma-k-nearest-neighbours-knn/* diakses pada 08 Desember 2018