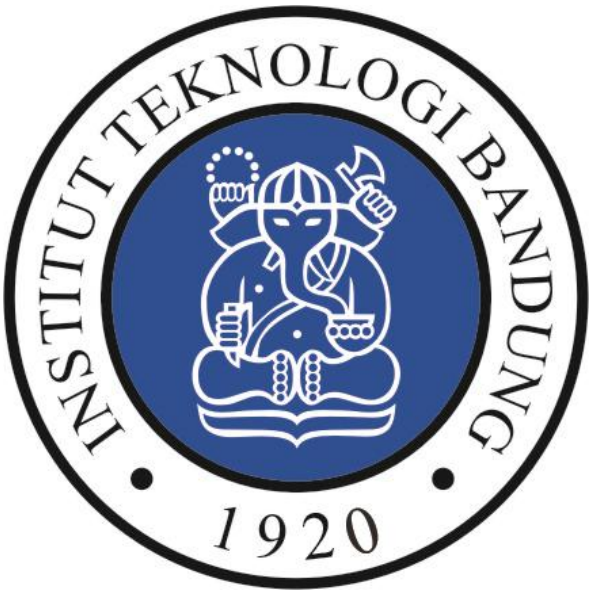
Laporan Implementasi Naïve Bayes dan Feed Forward Neural Network untuk Klasifikasi



dibuat oleh 707 Berusaha Imba:

13514002 – M. Diaztanto Haryaputra

13514023 – Fanda Yuliana Putri

13514025 – Ratnadira Widyasari

13514047 – Bervianto Leo P.

13514093 – Arnettha Septinez

Teknik Informatika

Institut Teknologi Bandung

Dasar Teori

1. Naïve Bayes

Naïve Bayes adalah metode pembelajaran dengan *supervised learning* (pembelajaran berlabel) dengan memanfaatkan probabilitas tiap atribut maupun kelas yang bertujuan mengklasifikasi data set. Langkah-langkah Naïve Bayes yaitu membuat frekuensi nilai setiap atribut lalu probabilitasnya. Setelah itu dapat melakukan proses klasifikasi dengan rumus

$P(v_j)$ *: probabilitas kelas* v_j*, dan*

$P(a_i|v_j)$ *: probabilitas atribut* a_i *pada* v_j

Untuk setiap nilai atribut ada probabilitas kelas masing-masing, seperti pada contoh di bawah.

outlook			temperature			humidity			windy			play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

Gambar 1, contoh model Naive Bayes

Misal ada contoh instance seperti berikut.

outlook	temp.	humidity	windy	play
sunny	cool	high	true	?

Gambar 2, contoh instance yg akan diklasifikasi

Maka perhitungan probabilitas kelas '*yes*' dan '*no*' adalah seperti gambar di bawah.

$$P(\text{play} = \text{yes}) = 9/14 \quad P(\text{play} = \text{no}) = 5/14$$

$$P(\text{yes})P(\text{sunny}|\text{yes})P(\text{cool}|\text{yes})P(\text{high}|\text{yes})P(\text{true}|\text{yes})$$
$$= 9/14 \cdot 2/9 \cdot 3/9 \cdot 3/9 \cdot 3/9 = 0.0053$$
$$P(\text{no})P(\text{sunny}|\text{no})P(\text{cool}|\text{no})P(\text{high}|\text{no})P(\text{true}|\text{no})$$
$$= 5/14 \cdot 3/5 \cdot 1/5 \cdot 4/5 \cdot 3/5 = 0.0206$$

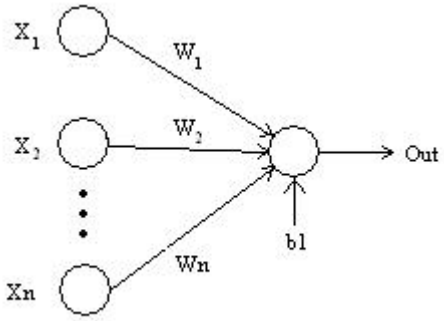
Gambar 3, perhitungan probabilitas tiap kelas

Dari dua probabilitas (karena ada dua kelas), diambil yang terbesar. Jadi, untuk contoh kali ini kelas Instance tersebut adalah '*no*'.

2. Feed Forward Neural Network (FFNN)

FFNN adalah sebuah metode pembelajaran mesin yang terinspirasi dari sistem saraf manusia. FFNN berbentuk menyerupai jaringan yang terhubung untuk menyalurkan informasi. FFNN terdiri dari *input node(s), output node(s),* dan *hidden layer* (jika ada).

2.1 Single-layer Perceptron



Gambar 1 Single-layer Perceptron

Single-layer Perceptron merupakan jenis FFNN yang terdiri dari 1 *single-layer output node*. Dalam jenis ini, masukan dari *input node* langsung diarahkan ke *output node* melalui *weights* yang acak. Dalam *output node,* beban dikali dengan masukan dan diolah dengan fungsi aktivasi sehingga menghasilkan output. Fungsi aktivasi yang biasa digunakan dalam FFNN adalah fungsi sigmoid.

Setelah dilakukan feed-forward processing, proses pembelajaran selanjutnya bisa dilakukan dengan menggunakan algoritma backpropagation. Dalam pembelajaran backpropagation, dilakukan penghitungan error untuk setiap *node* dalam jaringan. Untuk fungsi sigmoid, nilai yang digunakan merupakan turunan dari fungsi sigmoid sendiri.

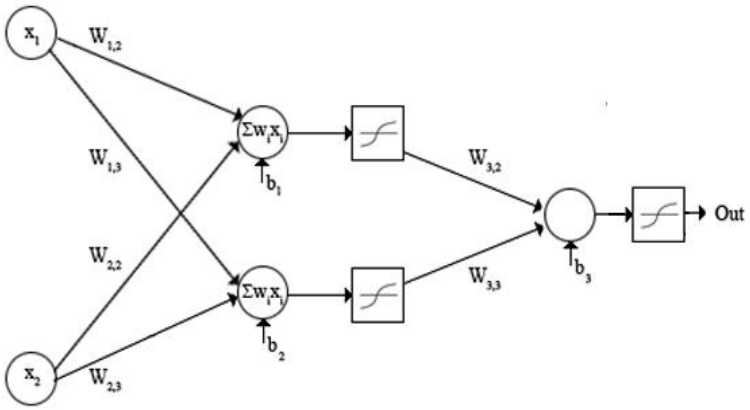
$$f(x) = \frac{1}{1 + e^{-x}}$$

Gambar 2 Fungsi Aktivasi Sigmoid

$$f'(x) = f(x)(1 - f(x)).$$

Gambar 3 Backpropagation

2.2 Multi-layer Perceptron



Gambar 2 Multi-layer Perceptron

Multi-layer perceptron merupakan suatu jaringan yang terdiri dari beberapa *node hidden(s)* dan *node output(s).* Untuk struktur lain dalam multi-layer perceptron sama dengan dalam single-layer perceptron, termasuk fungsi aktivasi sigmoid dan algoritma pembelajaran dengan menggunakan backpropagation.

Implementasi

1. **Naïve Bayes**

   Implementasi algoritma Naive Bayes kami ditaruh pada suatu kelas yang diturunkan dari *AbstractClassifier* dari Weka. Ada 4 prosedur yang di-*override*, yaitu *buildClassifier*, *distributionForInstance*, dan *classifyInstance*. Parameter yang tersedia untuk *classifier* ini adalah tipe filter yang digunakan, dengan dua pilihan, *Discretize* atau *NominalToNumeric*. Berikut dijelaskan secara singkat untuk setiap prosedur yang ada di *classifier* ini.

   1.1 buildClassifier(Instances data)

Prosedur ini mengisi data *classifier* yang berupa array 3 dimensi berisi probabilitas setiap kelas untuk setiap nilai pada setiap atribut. Awalnya dibangun terlebih dahulu array yang dibutuhkan, dengan array paling luar mewakili atribut, array berikutnya mewakili setiap nilai pada setiap atribut, dan array terdalam mewakili kelas untuk setiap nilai pada setiap atribut. Kemudian dihitung jumlah kemunculan setiap kelas untuk setiap nilai pada setiap atribut, lalu dibandingkan dengan jumlah total kemunculan setiap kelas untuk setiap atribut saja. Semuanya menggunakan iterasi konvensional.

1.2 distributionForInstance(Instance data)

Prosedur ini melakukan iterasi untuk menghitung probabilitas setiap kelas dari setiap nilai atribut parameter *data* untuk memenuhi rumus yang ada di dasar teori. Hasil perhitungan probabilitas setiap kelas disimpan dalam array bertipe *double* dan dikembalikan kepada pemanggil prosedur ini.

1.3 classifyInstance(Instance data)

Prosedur ini simpelnya memanggil prosedur *distributionForInstance(data)*. Lalu dicari nilai tertinggi dalam array bertipe *double* yang dikembalikan prosedur tadi, dan kemudian indeks tertingginya akan dikembalikan pada pemanggil prosedur ini.

2. Feed Forward Neural Network

Untuk implementasi algoritma FFNN, kami membuat satu kelas dengan beberapa fungsi dan prosedur. Kelas yang kami buat meng-*extends* AbstractClassifier milik Weka. Dalam salah satu prosedur yang kami implementasikan, terdapat 2 prosedur yang meng-*override* prosedur milik Weka, yaitu buildClassifier dan classifyInstance. Untuk beberapa *variable* yang digunakan dalam algoritma ini, seperti jumlah iterasi, jumlah node dalam hidden layer, jumlah hidden layer, dll, merupakan masukan dari pengguna. Berikut merupakan deskripsi singkat untuk beberapa prosedur penting dalam implementasi kelas FFNN ini.

2.1 activationFunction (double sum)

Fungsi ini dugunakan untuk menghitung hasil penjumlahan dari hasil kali bobot masing-masing sisi dengan node yang berhubungan. Untuk perhitungan yang dilakukan, rumus yang digunakan adalah rumus untuk fungsi aktivasi sigmoid yang sama seperti yang dijelaskan di dasar teori. Bobot yang dihitunga dalam algoritma ini bisa dari hasil random (jika di awal pembelajaran) atau bobot hasil perhitungan *learning.*

2.2 feedForward(Instance instance)

Algoritma ini melakukan penghitungan terhadap input sehingga didapatkan output yang merupakan hasil dari penjumlahan dari perkalian bobot dengan angka dalam *input node.*

2.3 updateWeight(double[] tar)

Fungsi ini digunakan untuk melakukan pembelajaran dalam single-layer, pembelajaran dilakukan dengan memperbaharui bobot dari setiap *input node* yang ada. Pembelajaran untuk FFNN yang tidak memiliki hidden layer sengaja dibuat berbeda dengan yang memiliki hidden layer karena perhitungan yang dilakukan berbeda.

2.4 backPropagation(double[] tar)

Algoritma pembelajaran dengan backpropagation dilakukan untuk FFNN yang mempunyai hidden layer. Untuk algoritma ini, dilakukan juga penghitungan error terhadap output dan hidden layer. Setelah dilakukan penghitungan terhadap error kedua layer, proses selanjutnya adalah update bobot yang terdapat pada setiap sisi yang ada pada jaringan.

2.5 buildClassifier(Instances data) (@Override)

Secara umum, build classifier merupakan prosedur untuk melakukan pembelajaran terhadap data masukan, dengan hasil keluaran berupa model pembelajaran. Pembelajaran dilakukan secara berulang-ulang dengan menerapkan feedForward untuk setiap instance pada data dilajutkan dengan algoritma backpropagation atau updateWeight. Iterasi dilakukan terus-menerus sampai mendekati nilai threshold atau memenuhi jumlah iterasi.

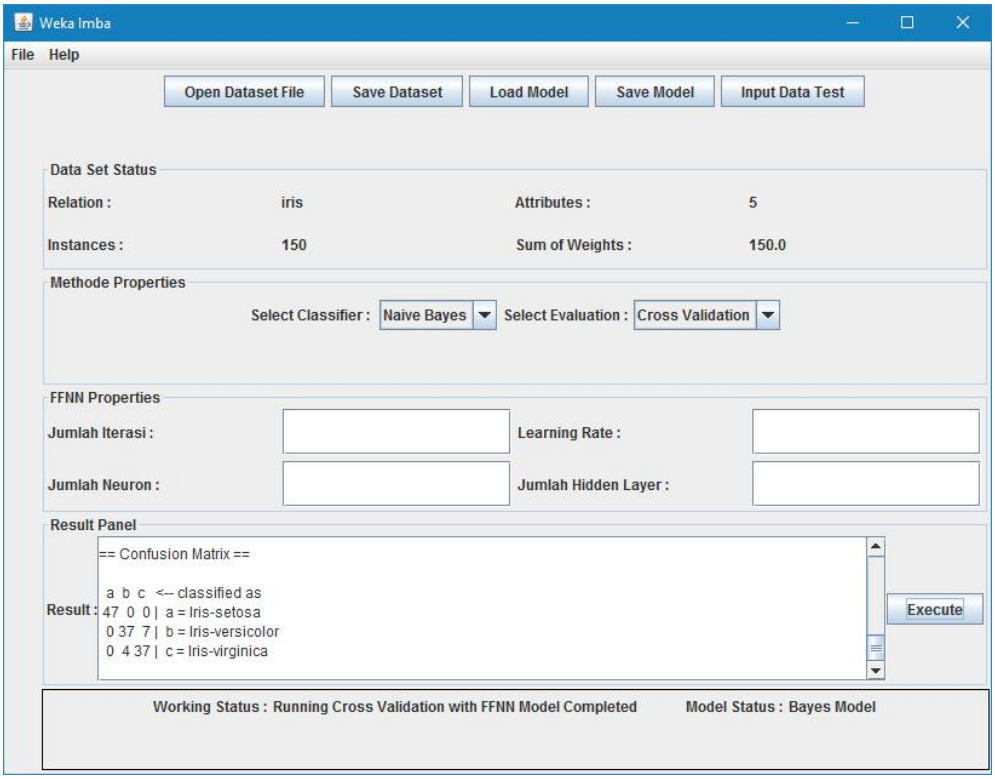
2.6 classifyInstance(Instance instance) (@Override)

Fungsi ini mengklasifikasi instance kedalam kelas-kelas yang ada sesuai dengan model yang sudah terbuat.

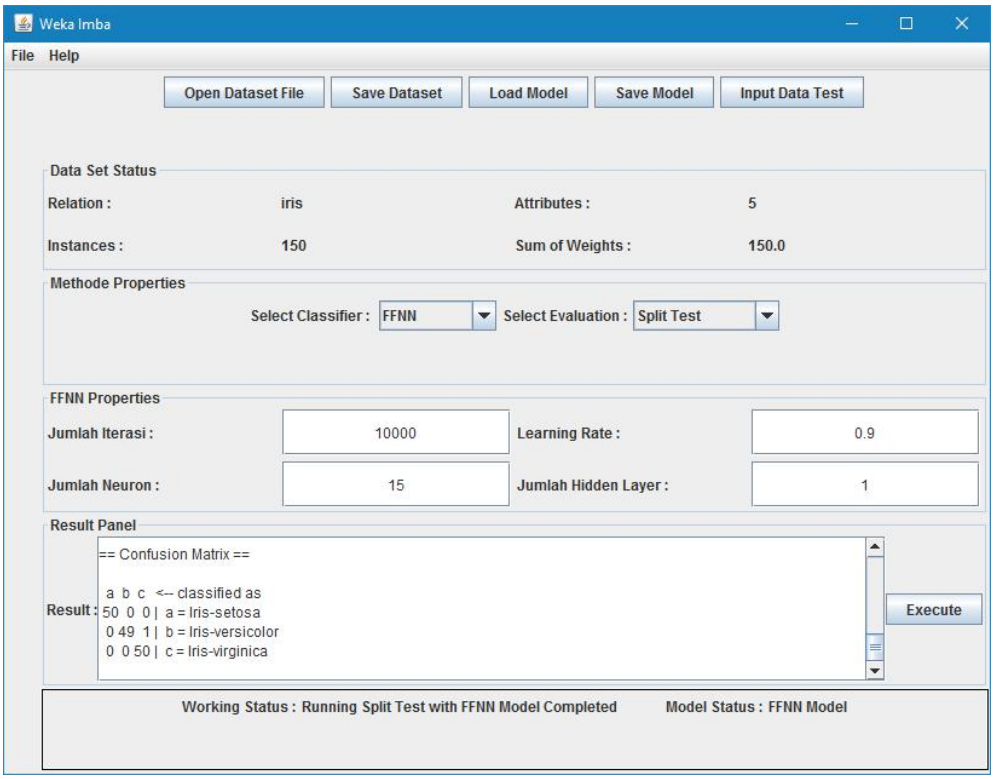
Hasil Implementasi

Percobaan pada data set iris.arff.

1. Naïve Bayes



2. Feed Forward Neural Network



Simpulan

FFNN metode pembelajaran terbaik dibandingkan dengan Naive Bayes untuk data set iris.arff. Berarti FFNN akan baik pada data set yang numeric sedangkan Naive Bayes akan baik pada data set yang nominal.