BAB III

METODOLOGI PENELITIAN

3.1 Objek Penelitian

Penelitian ini menggunakan data mahasiswa strata satu Universitas Multimedia Nusantara pada angkatan 2007 hingga angkatan 2018 sebagai objek penelitian. Data mahasiswa yang digunakan meliputi tahun masuk, program studi, tempat lahir, tahun lahir, jenis kelamin, *range* pendapatan orang tua, jalur masuk, nilai indeks prestasi semester 1 hingga semester akhir, jumlah satuan kredit semester 1 hingga semester terakhir, status apakah mendapat beasiswa atau tidak pada semester 1 hingga semester terakhir, dan status mahasiswa (*churn* atau *not churn*).

3.2 Metode Penelitian

Penelitian ini akan membangun model untuk mengklasifikasi mahasiswa ke dalam dua label yang berbeda, yaitu *churn* atau *not churn* dengan menggunakan algoritma jaringan saraf tiruan. Lima tahapan utama dalam penelitian ini dapat dilihat pada Gambar 3.1.

Gambar 3. 1 Tahapan Utama Metode Peneltian

3.2.1 Pengumpulan Data

Pengumpulan data dilakukan dengan meminta data secara langsung kepada pihak IT Universitas Multimedia Nusantara dengan menyerahkan surat ijin dari Program Studi Sistem Informasi. Data dengan ekstensi .xls tersebut berisi data mahasiswa strata satu yang terdaftar pada tahun 2007 hingga tahun 2018. Atribut data mahasiswa yang diberikan oleh divisi IT Universitas Multimedia Nusantara dapat dilihat pada Tabel 3.1.

Tabel 3. 1 Keterangan Atribut Data

Nama Atribut	Keterangan
TAHUN_MASUK	Tahun masuk mahasiswa
ACAD_PROG_PRIMARY	Kode program studi
PRODI	Nama program studi
FAKULTAS	Nama fakultas
BIRTHPLACE	Tempat lahir
CITY	Kota domisili
BIRTHYEAR	Tahun lahir
SEX	Jenis kelamin
TERM	Penanda semester untuk atribut IPS, SKS, beasiswa, dan surat peringatan
IPS	Nilai indeks prestasi semester pada semester 1 hingga semester terakhir
SKS	Jumlah satuan kredit semester pada semester 1 hingga semester terakhir
BEASISWA	Status mendapatkan beasiswa (Y/N) pada semester 1 hingga semester terakhir
SURAT_PERINGATAN	Status mendapatkan surat peringatan (Y/N) pada semester 1 hingga semester terakhir
STATUS_PERNIKAHAN	Status pernikahan mahasiswa
PENDAPATAN_ORTU	*Range* pendapatan orang tua
JALUR_MASUK	Jalur masuk (beasiswa, beasiswa *test,* prestasi, *regular test*)
STATUS	Indikasi mahasiswa *churn* atau *not churn*

3.2.2 *Data Preprocessing*

Data yang didapatkan tidak bisa secara langsung digunakan sebagai *input* karena masih ada data yang belum sesuai sehingga perlu dilakukan pra-pemrosesan data. Pada tahap ini akan dilakukan seleksi atribut, penanganan terhadap *missing value*, *reshape data*, *feature mapping*, dan menghapus data duplikat. *Tools* yang digunakan pada tahap *data preprocessing* adalah R dan Microsoft Excel. R digunakan untuk memanfaatkan library *reshape* dan Microsoft Excel ditujukan untuk tahap yang dilakukan secara manual.

3.2.2.1 Seleksi Atribut

Data mahasiswa yang didapatkan tidak dapat digunakan semua karena ada beberapa atribut yang datanya bersifat tumpang tindih, tidak tersedia karena belum dikelola oleh sistem, dan tidak mewakilkan *instance*. Oleh karena itu, atribut-atribut dari data harus diseleksi terlebih dahulu. Atribut dari data mahasiswa yang didapatkan dari pihak IT Universitas Multimedia Nusantara beserta keterangannya dapat dilihat pada Tabel 3.1 dalam subbab 3.2.1. Atribut dari data tersebut meliputi tahun_masuk, acad_prog_primary, prodi, fakultas, birthplace, city, birthyear, sex, term, ips, sks, beasiswa, surat_peringatan, status_pernikahan, pendapatan_ortu, jalur_masuk, dan status.

Pada data tersebut ada lima atribut yang harus dihilangkan, yaitu acad_prog_primary, fakultas, surat_peringatan, term, dan status_pernikahan. Berikut ini merupakan alasan mengapa atribut-atribut tersebut harus dihilangkan:

1. Atribut acad_prog_primary, atribut ini berisi kode prodi dari setiap mahasiswa. Atribut ini harus dihilangkan karena bersifat tumpang tindih dengan atribut prodi yang berisi nama program studi.
2. Atribut fakultas, atribut ini berisi nama fakultas dimana mahasiswa berada. Atribut ini juga dihilangkan karena bersifat tumpang tindih dengan atribut prodi. Atribut prodi dinilai sudah cukup mewakilkan mahasiswa karena bersifat lebih unik dibandingkan atribut fakultas.
3. Atribut surat_peringatan, atribut ini berisi status (Y/N) apakah mahasiswa mendapatkan surat peringatan pada semester 1 hingga semester terakhir. Atribut ini harus dihilangkan karena datanya tidak tersedia dikarenakan data tersebut belum dikelola oleh sistem.
4. Atribut status_pernikahan, atribut ini dihilangkan karena semua mahasiswa memiliki status pernikahan yang sama sehingga atribut ini tidak berkontribusi dalam klasifikasi.
5. Atribut term, atribut ini merupakan penanda semester untuk atribut IPS, SKS, beasiswa, dan surat peringatan. Atribut ini akan dihilangkan tapi sebelumnya akan digunakan terlebih dahulu pada tahap *data preprocessing.*

3.2.2.2 *Missing Value*

Missing value adalah ketidaktersediaan data atau informasi yang diperoleh karena suatu alasan tertentu. Hal ini bisa terjadi karena data yang kurang lengkap, data tersebut sulit dicari, atau data memang tidak tersedia pada *database* perusahaan

sehingga penyedia data tidak dapat memberikan data secara lengkap. Penanganan terhadap *missing value* akan dilakukan dengan cara mengisi *value* yang kosong dengan -1. Pemberian nilai -1 ini dilakukan karena ada beberapa atribut yang memiliki data bernilai nol sehingga harus ada nilai yang membedakan antara data yang bernilai kosong (*null*) dan data yang bernilai nol.

3.2.2.3 *Reshape Data*

Pada tahap ini akan dilakukan perubahan posisi data pada atribut IPS, SKS dan Beasiswa dari vertikal menjadi horisontal sehingga satu mahasiswa hanya akan memiliki satu baris *record*. Hal ini dilakukan agar data dapat lebih mudah dibaca dan lebih mudah diimplementasi ke jaringan saraf tiruan. Hal pertama yang dilakukan adalah menambah kolom pada data. Kolom ini akan diisi *flag* untuk IPS, SKS dan Beasiswa berdasarkan data dari atribut term. Kemudian, menggunakan *library reshape* yang tersedia pada R dengan memanfaatkan kolom *flag* yang sebelumnya dibuat pada Excel.

3.2.2.4 *Feature Mapping*

Semua data dengan tipe *non numeric* akan diubah menjadi tipe *numeric*. Data tersebut adalah data yang berada pada atribut prodi, tempat_lahir, tempat_tinggal, jenis_kelamin, beasiswa, pendapatan_ortu, jalur_masuk, dan status. Hal ini dilakukan karena algoritma jaringan saraf tiruan hanya bisa menerima input berupa data yang bertipe numerik atau angka dan menghasilkan output yang berupa numerik juga. *Feature mapping* dilakukan dengan mengubah

isi atribut secara berurutan sesuai dengan jumlah kategorinya. Jumlah kategori pada atribut yang akan di-*mapping* dapat dilihat pada Tabel 3.2.

Tabel 3. 2 Atribut dan Jumlah Kategori Pada Data

Nama Atribut	Jumlah
PRODI	13
BIRTHPLACE	477
CITY	266
SEX	2
PENDAPATAN_ORTU	6
JALUR_MASUK	4
BEASISWA	2
STATUS	2

3.2.2.5 Menghapus Data Duplikat

Pada data mahasiswa masih terdapat beberapa data yang kembar atau duplikat. Misalnya, satu mahasiswa memiliki tiga *record* data dimana seharusnya satu mahasiswa hanya memiliki satu *record* data saja. Oleh karena itu, data mahasiswa yang duplikat ini akan dihapus agar data yang digunakan dalam proses *training* dan *testing* dapat memberikan hasil yang lebih akurat.

3.2.3 Membangun Model

Model klasifikasi jaringan saraf tiruan (JST) dibangun dengan menggunakan *library neuralnet*. Terdapat tiga lapisan pada model klasifikasi jaringan saraf tiruan, yaitu *input layer, hidden layer*, dan *output layer*. Pada model JST ini akan menggunakan satu *hidden layer*. Jaringan saraf tiruan dengan satu *hidden layer* dapat memperkirakan fungsi apa pun yang berisi pemetaan terus menerus dari satu ruang terbatas hingga lainnya (Budiharto, Machine Learning dan

Computational Intelligence, 2016). Faktanya, untuk banyak *practical problem*, tidak ada alasan untuk menggunakan lebih dari satu *hidden layer* (Heaton, 2008).

Di dalam *hidden layer* (lapisan tersembunyi) terdapat sejumlah neuron. Salah satu metode *rule of thumb* untuk menentukan jumlah neuron yang tepat untuk digunakan dalam lapisan tersembunyi adalah jumlah neuron pada lapisan tersembunyi harus kurang dari dua kali ukuran lapisan *input* (Heaton, 2008). Menurut Heaton (2008), menggunakan terlalu banyak neuron pada lapisan tersembunyi dapat menyebabkan beberapa masalah. Pertama, terlalu banyak neuron di *hidden layer* dapat menyebabkan *overfitting*. *Overfitting* terjadi ketika jaringan saraf memiliki begitu banyak kapasitas pemrosesan informasi sehingga keterbatasan informasi yang terkandung dalam *training set* tidak cukup untuk melatih semua *neuron* yang terdapat pada *hidden layer* (Heaton, 2008). Masalah kedua, banyaknya jumlah *neuron* dalam *hidden layer* dapat meningkatkan waktu yang dibutuhkan untuk melatih jaringan. Jumlah waktu pelatihan dapat meningkat ke titik yang tidak mungkin untuk melatih jaringan saraf secara memadai (Heaton, 2008). Oleh karena itu, pemilihan jumlah *hidden layer* dan *neuron* pada *hidden layer* yang tepat akan mempersingkat waktu *training* dan menghindari terjadinya *overfitting*.

Arsitektur jaringan saraf tiruan ini memiliki 1 *input layer,* 1 *hidden layer,* dan 1 *output layer.* Pada model jaringan saraf tiruan ini akan dilakukan pengujian terlebih dahulu terhadap jumlah neuron pada *hidden layer* untuk menemukan

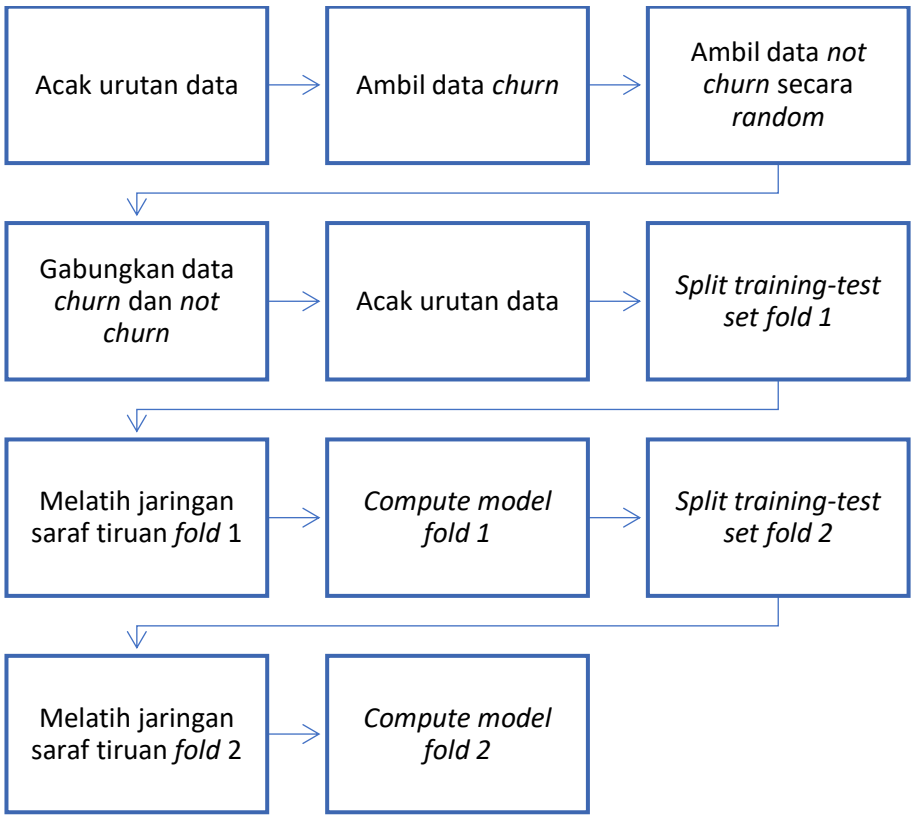
jumlah *hidden neuron* terbaik. *Input layer* model ini memiliki 50 neuron, sedangkan pada *output layer* memiliki 1 neuron dengan dua label, yaitu *churn* atau *not churn*.

Pada *hidden layer*, jumlah *hidden neuron* akan diuji terlebih dahulu sehingga dapat ditentukan jumlah *hidden neuron* terbaik. *Hidden neuron* yang diuji adalah berjumlah 15, 20, dan 25. Setelah menentukan jumlah *hidden neuron* terbaik, akan dilanjutkan dengan memilih *learning rate* terbaik. *Learning rate* yang diuji adalah 0,1, 0,01, dan 0,001. Parameter lainnya yang digunakan pada model dengan *library neuralnet* adalah *threshold*, *stepmax, startweights,* dan *act.fct*. *Threshold* yang digunakan adalah 0,01. *Stepmax* yang digunakan adalah 1.000.000. *Startweights* yang digunakan adalah 0,001. *Act.fct* adalah fungsi aktivasi. Model jaringan saraf tiruan ini akan menggunakan fungsi sigmoid sebagai fungsi aktivasinya.

Variabel prediktor (*input*) yang digunakan adalah tahun masuk, program studi, tempat lahir, tahun lahir, jenis kelamin, *range* pendapatan orang tua, jalur masuk, nilai indeks prestasi semester 1 hingga semester akhir, jumlah satuan kredit semester 1 hingga semester terakhir, dan status apakah mendapat beasiswa atau tidak pada semester 1 hingga semester terakhir. Variabel target (*output*) adalah status *churn* atau tidak *churn*. Penjelasan lebih lanjut tentang *input* dan *output* yang telah disebutkan dapat dilihat pada Tabel 4.2 (subbab 4.2) yang merupakan hasil akhir dari *data preprocessing*.

3.2.4 Pelatihan Model

Tahap ini dilakukan dengan melatih sebuah model sebanyak 10 kali dengan parameter yang sama dan menggunakan data mahasiswa *not churn* yang berbeda yang akan diambil secara *random*. Dilatih sebanyak 10 kali karena populasi mahasiswa *not churn* berjumlah 13.482 dan jumlah sampel mahasiswa *not churn* yang diambil dalam sekali latih adalah 2.912. Dengan demikian, jumlah minimal pelatihan model yang harus dicapai adalah sebanyak 5 kali. Oleh karena itu, untuk mendapatan kualitas model yang lebih *valid* dan objektif, pada penelitian ini akan dilakukan sebanyak 10 kali. Urutan langkah-langkah dalam pelatihan model dapat dilihat pada Gambar 3.2.



Gambar 3. 2 Langkah Proses Pelatihan Model

Rata-rata akurasi 2-*fold* dari 10 kali pelatihan akan dihitung dan dijadikan sebagai nilai akurasi model. Setelah itu, rata-rata dari akurasi setiap model akan dihitung. Berikut ini merupakan 11 langkah yang dilakukan saat pelatihan model berdasarkan gambar alur proses di atas:

1. Acak urutan *original dataset*, hal ini dilakukan agar data dapat teracak sehingga dapat mengambil data yang *random* pada langkah selanjutnya.
2. Mengambil semua data mahasiswa *churn* saja dan ditampung ke dalam suatu variabel, contoh: x.
3. Mengambil sampel mahasiswa *not churn* sejumlah mahasiswa *churn* secara *random* dan ditampung ke dalam suatu variabel, contoh: y. Hal ini dilakukan untuk membuat *dataset* yang seimbang. Metode menyeimbangkan data untuk membuat *churn classifier* ini didukung dan dilakukan pada penelitian Aulck, Velagapudi, Blumenstock, & West (2016) dan Delen, D. (2010).
4. Menggabungkan data mahasiswa *churn* (x) dan *not churn* (y) menjadi sebuah *dataset* baru.
5. Mengacak urutan *dataset* tersebut agar mendapatkan data yang *random* saat melakukan *split training-testing set.*
6. Membagi *dataset* untuk *training set* dan *testing set*. *Training set* diambil dari data ke-1 hingga data ke-2912 dan *testing set* diambil dari data ke-2913 hingga data ke-5824.
7. Melatih model jaringan saraf tiruan *fold* pertama dengan menggunakan *data training*.

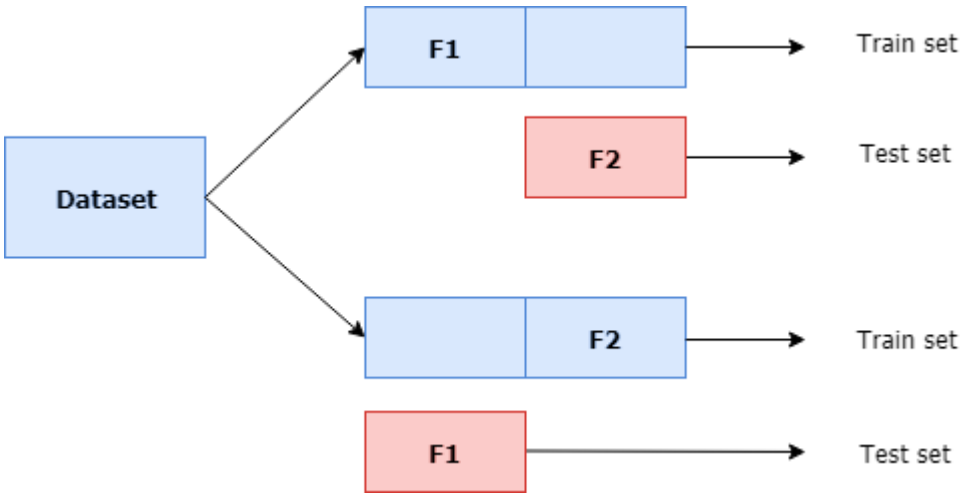
8. Melakukan *compute* model *fold* pertama untuk memprediksi dengan *data testing*.
9. Membagi *dataset* untuk *training set* dan *testing set*. *Training set* diambil dari data ke-2913 hingga data ke-5824 dan *testing set* diambil dari data ke-1 hingga data ke-2912.
10. Melatih model jaringan saraf tiruan *fold* kedua dengan menggunakan *data training*.
11. Melakukan *compute* model *fold* kedua untuk memprediksi dengan *data testing*.

3.2.5 Evaluasi Kinerja Model

Penelitian ini menggunakan metode *confusion matrix* (lihat pada subbab 2.5) untuk menghitung nilai akurasi, *precision, recall*, dan *F1 score*. Nilai akurasi menunjukkan seberapa besar tingkat pengenalan model klasifikasi terhadap setiap *tuple*. *Precision* akan menghitung seberapa banyak data yang diberi label positif dan benar-benar positif. *Recall* akan mengukur seberapa banyak label positif yang diklasifikasi dengan benar. *F1 score* akan menggabungkan nilai *precision* dan *recall* sehingga menghasilkan rata-rata harmonik.

Penelitian ini akan menggunakan teknik *2-fold cross validation* (lihat pada subbab 2.4) sehingga memiliki 2 iterasi. Ilustrasi metode *2-fold cross validation* dapat dilihat pada Gambar 3.3. Pada iterasi pertama, *fold* pertama akan digunakan sebagai data latih dan *fold* kedua sebagai data uji. Lalu, pada iterasi kedua, *fold*

pertama sebagai data uji dan *fold* kedua sebagai data latih. Setiap iterasi tersebut akan dihitung hasilnya sehingga menghasilkan nilai akurasi, *precision, recall,* dan *F1 score* secara keseluruhan dari model. Setelah itu, hasil keseluruhan evaluasi dari setiap iterasi tersebut akan dirata-ratakan dan digunakan sebagai hasil evaluasi model per pelatihan (10 kali). Kemudian hasil evaluasi model per pelatihan akan dikalkulasi sehingga hasil tersebut akan menjadi hasil evaluasi model secara keseluruhan.



Gambar 3. 3 Ilustrasi *2-Fold Cross Validation*