BAB 3
METODOLOGI PENELITIAN

3.1 Metodologi Penelitian

Pada bagian ini akan dijelaskan langkah langkah yang dilakukan dalam melakukan penelitian. Langkah langkah tersebut adalah telaah literatur, pengumpulan data, pengolahan data, perancangan model, pengujian dan evaluasi model.

3.1.1 Studi Literatur

Pada tahap studi literatur, dilakukan pengumpulan materi atau literatur dari berbagai sumber seperti jurnal, artikel, dan sumber lainnya yang berkaitan dengan penelitian ini, serta dilakukan analisis terhadap hasil-hasil penelitian sebelumnya untuk memperoleh pemahaman yang lebih mendalam mengenai topik yang diteliti.

3.1.2 Pengumpulan Data

Data pada penelitian ini didapatkan dari *website* Kaggle. *Dataset* tumor otak dari Jun Cheng adalah kumpulan citra MRI T1-weighted yang diperoleh dengan kontras gadolinium, digunakan untuk klasifikasi tiga jenis tumor otak: *meningioma*, *glioma*, dan *pituitary tumor*. Dataset ini berisi total 3.064 gambar dalam format .mat, yang dikumpulkan dari 233 pasien di Nanfang Hospital, Guangzhou, China dan General Hospital, Tianjin Medical University, China. Gambar-gambar tersebut merupakan potongan *axial* 2D. Distribusi jumlah setiap jenis tumor otak dalam dataset yang digunakan ditampilkan pada Tabel 3.1.

Tabel 3.1. Distribusi *dataset* tumor otak dari Jun Cheng

Jenis Tumor	**Jumlah Gambar**
Meningioma	708
Glioma	1426
Pituitary Tumor	930
Total	**3064**

3.1.3 Pengolahan Data

Dataset yang digunakan pada penelitian ini aslinya memiliki format ekstensi *file* gambar berupa .mat, sehingga langkah pertama yang dilakukan adalah mengonversi seluruh *file* gambar menjadi format .jpg untuk mempermudah proses pemrosesan selanjutnya. Setelah dilakukan konversi, augmentasi data dijalankan guna meningkatkan representasi fitur tanpa menambah jumlah data asli dan agar jumlah data menjadi lebih seimbang. Hasil augmentasi kemudian digabungkan dengan data gambar asli, di mana penggabungan dilakukan dengan tetap mempertahankan label masing-masing gambar. Seluruh data yang telah digabungkan ini kemudian dimasukkan ke dalam InceptionV3 guna mendapatkan fitur numerik hasil representasi dari gambar. Fitur numerik tersebut disimpan dalam sebuah *file* berekstensi .txt. Selanjutnya, fitur-fitur ini diproses menggunakan algoritma *Bird Eye View* (BEV) untuk melakukan seleksi fitur. Fitur-fitur terpilih kemudian dibagi menjadi dua bagian dengan rasio 80:20, di mana 80% digunakan sebagai data pelatihan atau data *training* dan 20% sisanya sebagai data pengujian atau data *validation*.

Pada *dataset* yang digunakan dalam penelitian ini, setiap jenis tumor otak diberi label numerik. Label 1 digunakan untuk menandai gambar dengan jenis tumor *Meningioma*, label 2 untuk gambar dengan jenis tumor Glioma, dan label 3 untuk gambar dengan jenis tumor *Pituitary Tumor*.

3.1.4 Perancangan Model

Pada tahap perancangan model, algoritma *Extreme Gradient Boosting* (XGBoost) dipilih sebagai model klasifikasi. Untuk memperoleh performa terbaik dari model, dilakukan proses *Grid Search* guna mencari kombinasi *hyperparameter* yang optimal. Proses ini mencakup eksplorasi berbagai nilai dari sejumlah *hyperparameter* penting seperti kedalaman pohon, *learning rate*, jumlah estimator, dan regularisasi. Hasil pencarian akan menentukan konfigurasi terbaik yang digunakan dalam pelatihan akhir model. Daftar lengkap *hyperparameter* yang digunakan ditampilkan pada Tabel 3.2.

Hyperparameter yang digunakan
objective
num_class
max_depth
min_child_weight
learning_rate
n_estimators
subsample
colsample_bytree
gamma
reg_alpha
reg_lambda

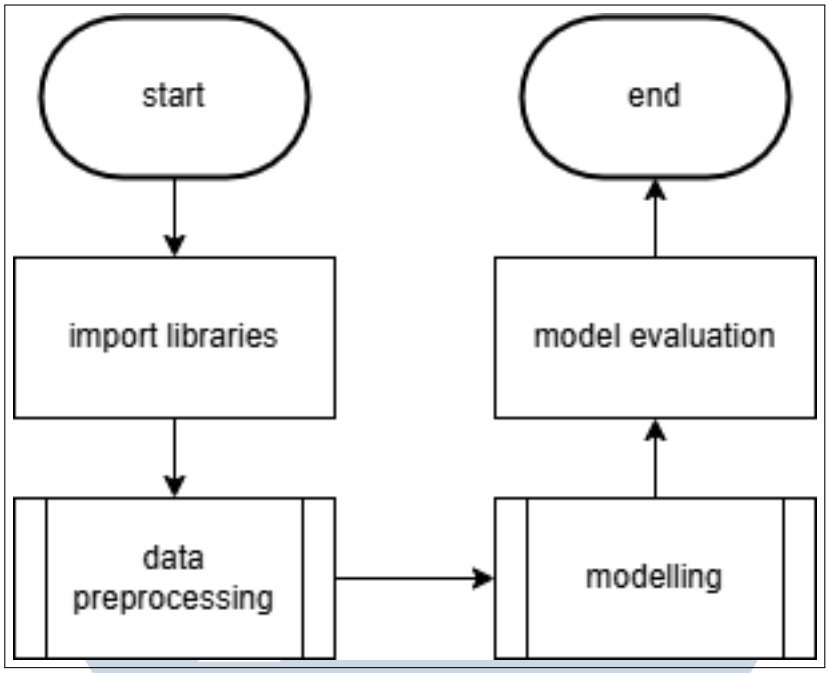
Tabel 3.2. *Hyperparameter* yang disesuaikan dalam proses Grid Search

3.1.5 Pengujian dan Evaluasi Model

Pengujian akan dilakukan menggunakan *confusion matrix* untuk menghitung metrik evaluasi berupa *accuracy*, *precision*, *recall*, dan F1-score. Jika hasil pengujian menunjukkan performa model yang belum optimal, maka akan dilakukan evaluasi untuk meningkatkan kualitas hasil.

3.2 Perancangan Sistem

Pada penelitian ini, sistem akan dibangun dalam beberapa tahapan, secara umum tahapan dapat dilihat pada Gambar 3.1.



Gambar 3.1. *Flowchart* tahapan umum pembangunan sistem

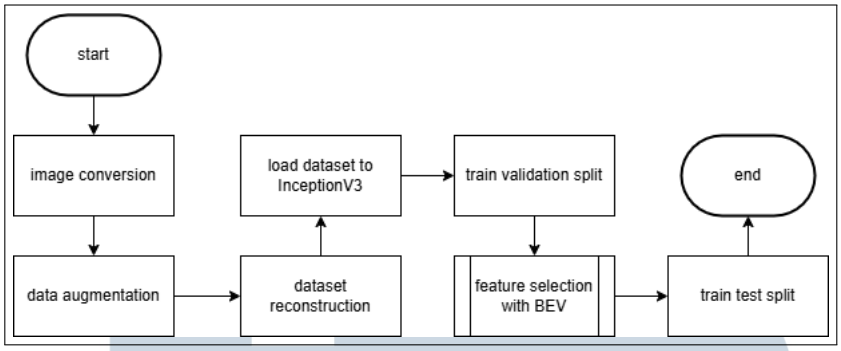
Proses dimulai dengan melakukan *import library*. Selanjutnya, dilakukan proses *data preprocessing* dilakukan untuk mengubah dan mempersiapkan data agar siap digunakan dalam proses *modelling*. Kemudian, tahap *modelling* dijalankan untuk membangun dan melatih model menggunakan algoritma XGBoost. Terakhir, dilakukan proses *model evaluation*, di mana performa model dinilai menggunakan metrik yang telah dipilih untuk memastikan keakuratan model.

3.2.1 Import Libraries

Pada tahapan ini, semua *library* yang akan digunakan dalam pembangunan sistem akan dimuat terlebih dahulu. Beberapa *library* yang akan digunakan adalah *numpy, shutil, tensorflow, keras*, dan os.

3.2.2 Data Preprocessing

Berikut adalah *flowchart* dari proses *data preprocessing*:

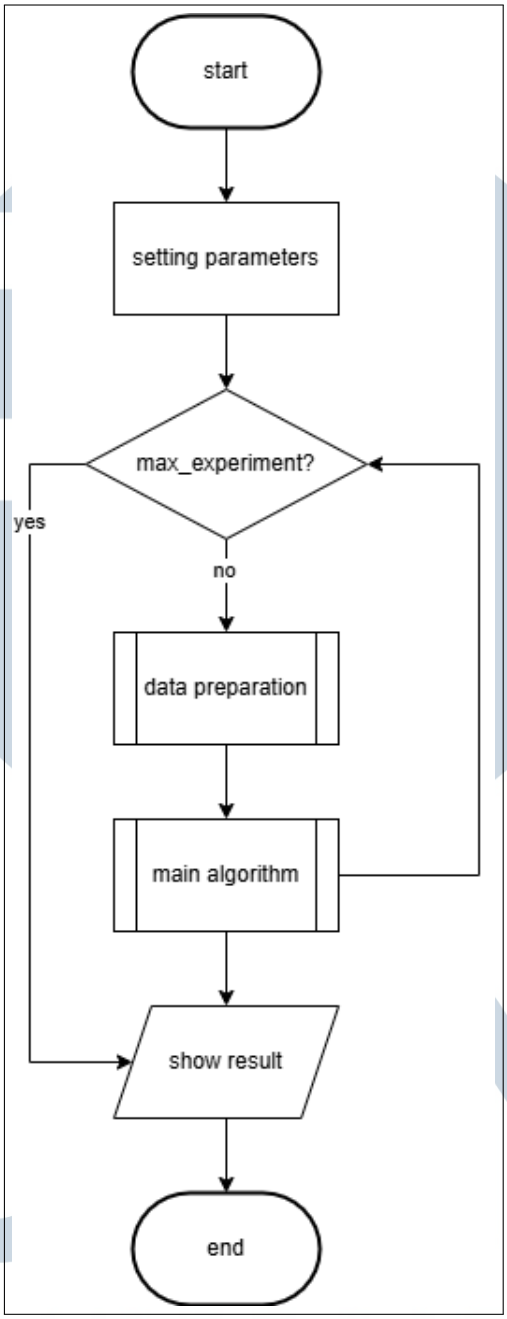


Gambar 3.2. *Flowchart data preprocessing*

Berdasarkan Gambar 3.2 proses dimulai dengan *image conversion*, yaitu proses konversi ekstensi file gambar dari format .mat ke format .jpg agar untuk mempermudah proses augmentasi data. Setelah gambar dikonversikan, proses dilanjutkan ke *data augmentation*, yang berguna untuk memperbesar jumlah data dan menyeimbangkan data dengan cara melakukan transformasi seperti rotasi, *flipping*, *zoom*, dan *shifting* terhadap gambar. Proses augmentasi dilakukan menggunakan *ImageDataGenerator* dari *TensorFlow* dan jumlah augmentasi disesuaikan berdasarkan kelas gambar.

Hasil augmentasi kemudian digabungkan kembali dengan dataset asli dalam proses *dataset reconstruction*, sehingga menghasilkan satu *dataset* baru yang berisikan gambar asli dan gambar hasil augmentasi. *Dataset* yang telah direkonstruksi ini kemudian dimasukkan ke dalam model InceptionV3 untuk memproyeksikan gambar menjadi representasi numerik (fitur).

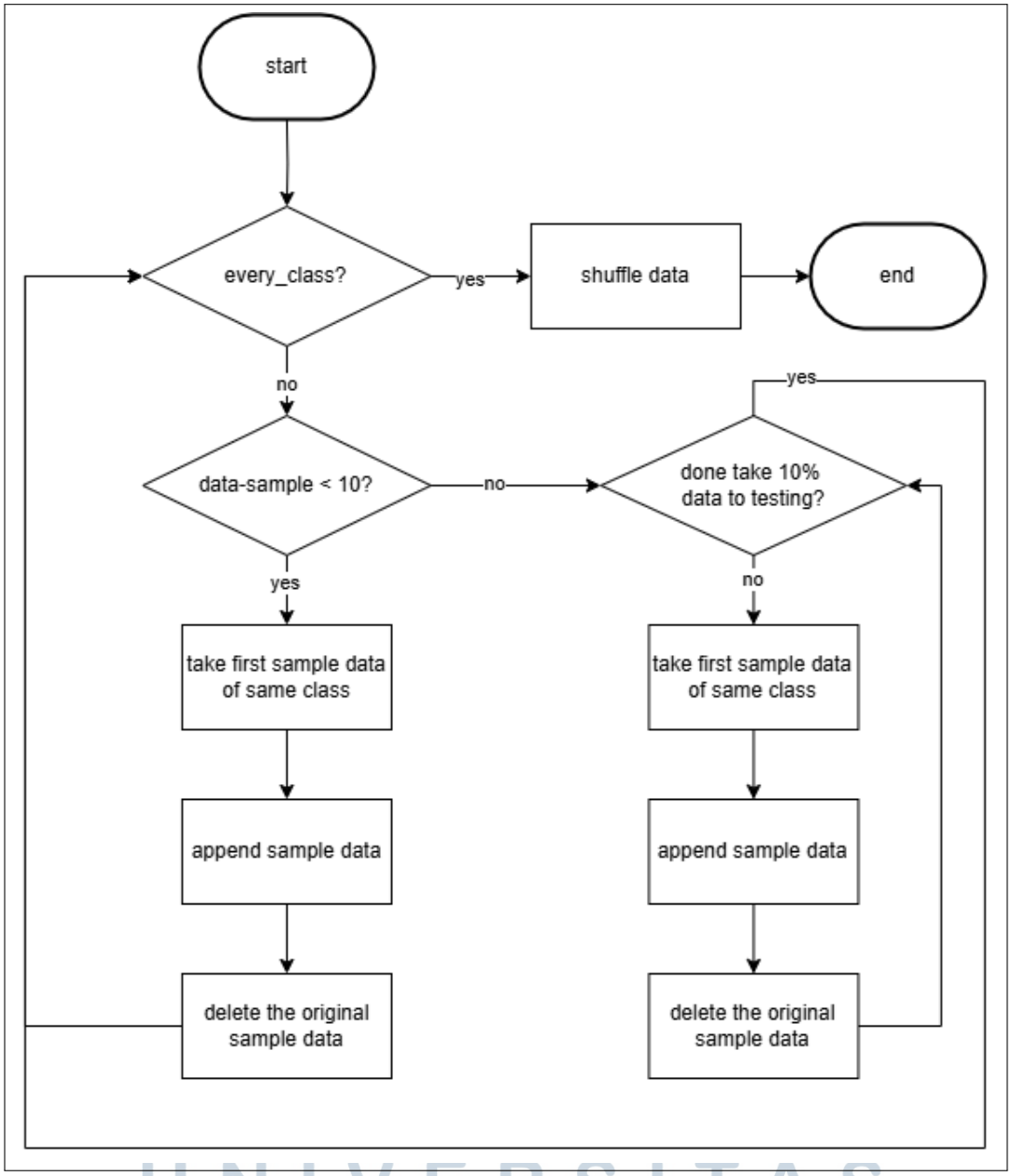
Setelah mendapat representasi numerik dari gambar atau fitur-fitur, proses dilanjutkan ke tahapan *train validation split*, proses ini dilakukan untuk memisahkan data *validation* dari data yang akan dimasukkan ke proses *feature selection*. Lalu proses berlanjut ke *feature selection with BEV*, *feature selection* yang digunakan adalah *Bird Eye View* (BEV). Metode ini digunakan untuk memilih fitur-fitur penting yang relevan, sehingga dapat mengurangi kompleksitas data dan meningkatkan performa model. Terakhir, proses *train validation split* diterapkan terhadap fitur yang terseleksi, fitur akan dibagi menjadi dua bagian yaitu 80% untuk data *training* dan 20% untuk data *validation*.



Gambar 3.3. *Flowchart feature selection with BEV*

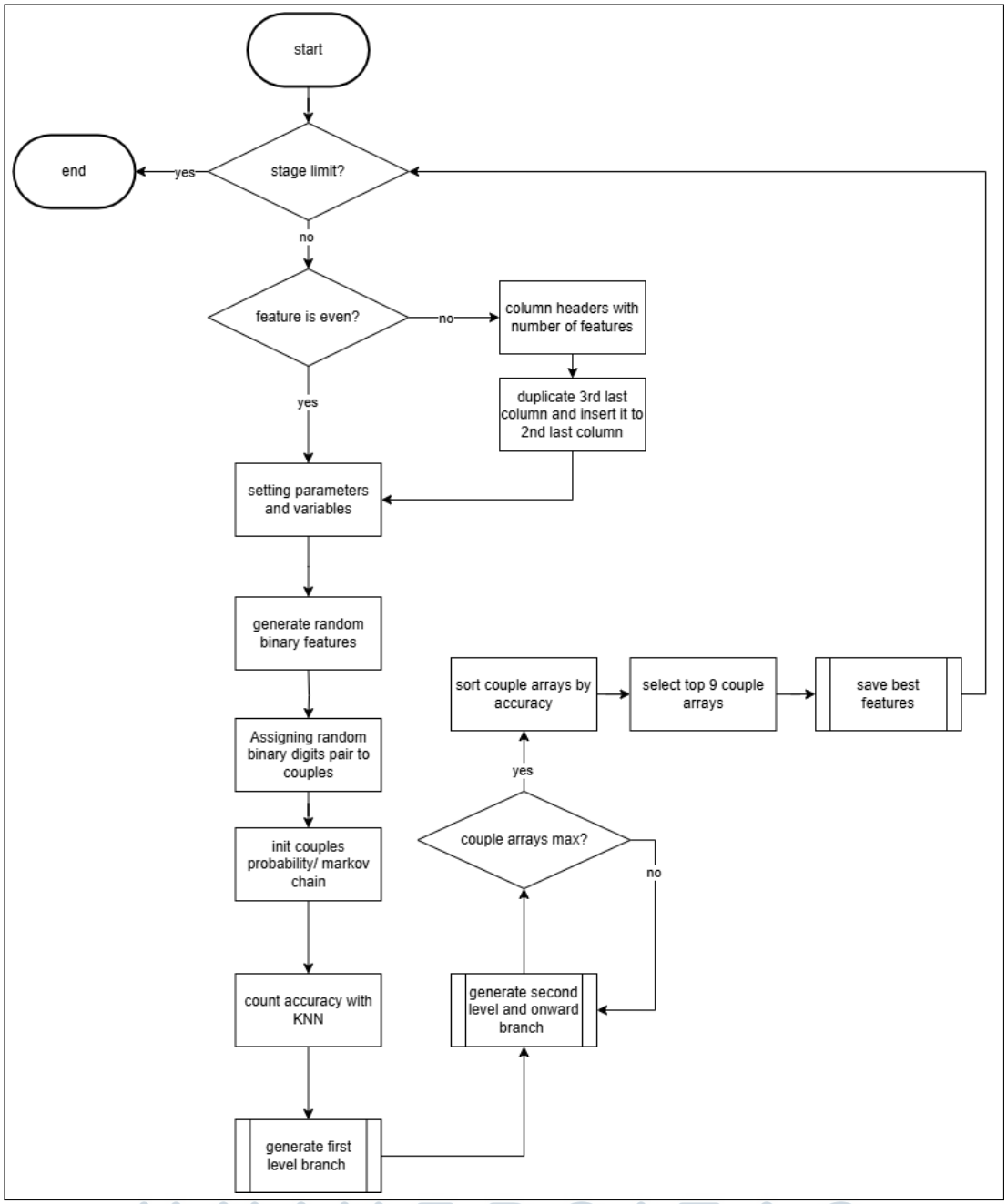
Proses BEV dapat dilihat pada Tabel 3.3. Tahapan BEV dimulai dari *setting parameters* seperti jumlah maksimal eksperimen, jumlah maksimal *branch* sebelum dipilih *top best branch*, dan jumlah *top best branch*. Lalu masuk ke proses *data preparation* guna *dataset* yang akan dimasukkan ke proses *main algorithm* sudah sesuai format. Pada proses *main algorithm* akan dilakukan *feature selection*, jika jumlah maksimal eksperimen sudah terpenuhi maka hasil dari *feature selection* akan

ditampilkan.



Gambar 3.4. *Flowchart data preparation* pada proses BEV

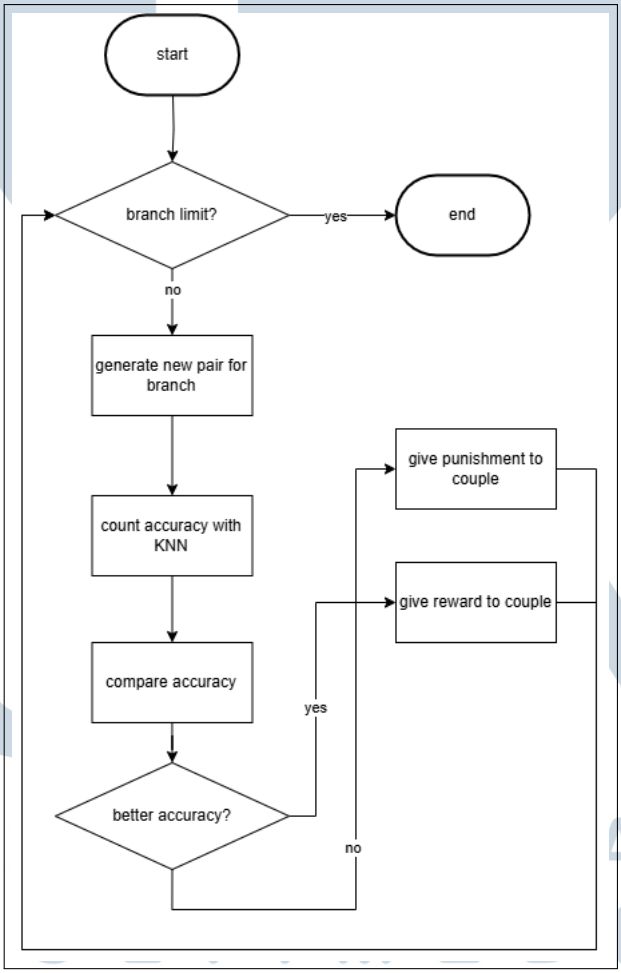
Berdasarkan Gambar 3.4 proses dimulai dengan pengecekan jumlah *sample* data per kelasnya. Jika jumlah *sample* data suatu kelas kurang dari 10 buah maka data pertama akan diambil dan ditaruh pada paling akhir dataset sebagai data *testing*. Jika Jika jumlah *sample* data suatu kelas lebih dari 10 buah maka 10% dari jumlah data kelas tersebut akan diambil guna dijadikan data *testing*.



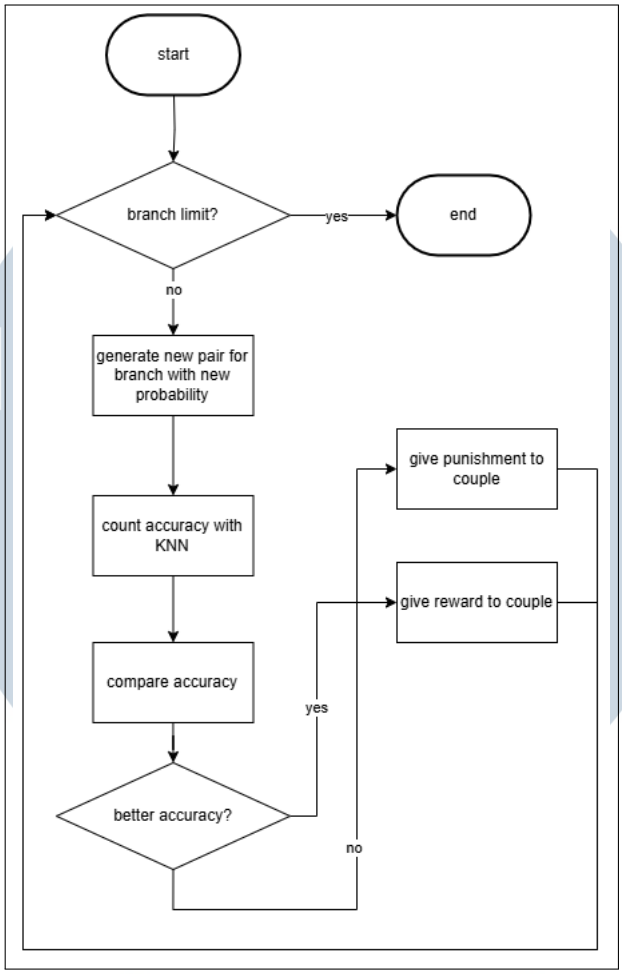
Gambar 3.5. *Flowchart main algorithm* pada proses BEV

Proses utama *feature selection* pada Tabel 3.5 dimulai dengan melakukan pengecekan apakah *stage* sekarang sudah melebihi batas atau belum. Jika belum maka proses akan dilanjutkan ke proses pengecekan apakah jumlah fitur berjumlah genap atau ganjil. Jika ganjil maka fitur kedua dari akhir akan disematkan ke kolom ketiga dari akhir atau disebelah kirinya. Setelah itu akan diatur parameter dan variabel yang berhubungan dengan pembuatan *branch*. Setelah itu fitur-fitur

tersebut akan diberikan *random binary* yang terdiri dari 0 dan 1. Jika fitur mendapat *bit* 0 maka fitur tersebut tidak akan terpakai, jika mendapat *bit* 1 maka fitur itu akan terpakai. Kemudian rangkaian representasi *binary* dari fitur-fitur akan dilakukan *coupling* atau *pairing*. Lalu *Couples* tersebut akan diberikan probabilitas berupa *Markov Chain*. Setelah itu rangkaian representasi *binary* fitur-fitur akan dihitung keakurasiannya dengan menggunakan KNN. Lalu proses berlanjut ke generasi *first level branch* dan seterusnya, jika jumlah *couple arrays* atau *branch* maksimal tercapai maka akan dipilih 9 *branch* yang memiliki akurasi tertinggi. Kemudian informasi mengenai *branch* tersebut akan disimpan. Proses akan berhenti jika *stage* sudah memenuhi batas maksimal.



Gambar 3.6. *Flowchart generate first level branch* pada proses BEV

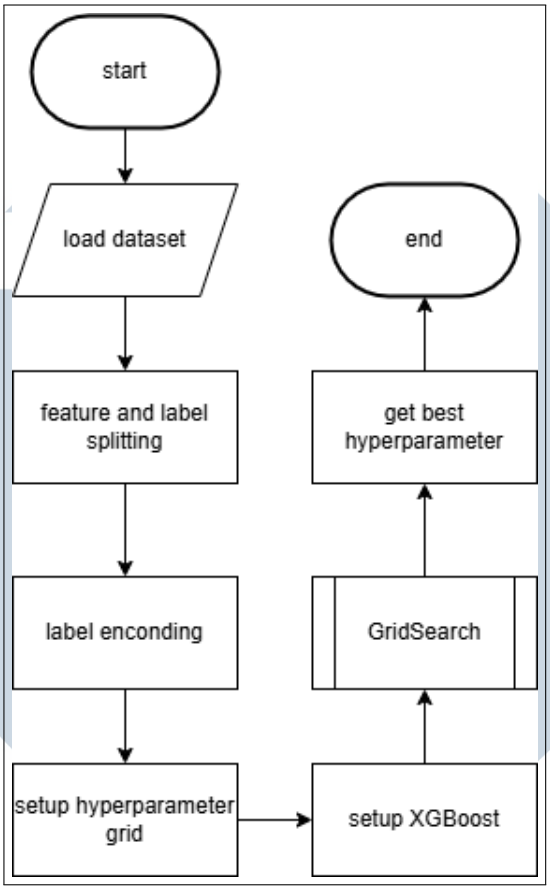


Gambar 3.7. *Flowchart generate second and onward level branch* pada proses BEV

Gambar 3.6 dan Gambar 3.7 menunjukkan tahapan generasi *branch*. Proses dimulai dari pengecekan apakah jumlah *branch* sudah maksimal. Jika belum maka akan dilakukan generasi *binary pair* baru pada *couples*. Kemudian rangkaian representasi *binary* fitur-fitur akan dihitung keakurasiannya dengan menggunakan KNN. Hukuman kepada *couples* jika hasil akurasi rangkaian baru lebih jelek dari sebelumnnya maka akan diberikan dan *reward* jika hasil akurasi rangkaian baru lebih baik. Perbedaan antara proses generasi *first branch* dan *second level and onward branch* adalah pada proses generasi *binary pair* baru.

3.2.3 Modelling

Berikut adalah *flowchart* dari proses *modelling*:



Gambar 3.8. Flowchart *modelling*

Berdasarkan Gambar 4.17 proses dimulai dengan *load dataset*, *dataset* diambil dari hasil *data preprocessing* yang telah melalui tahap *feature selection*. Setelah itu dilakukan *feature* and *label splitting*.

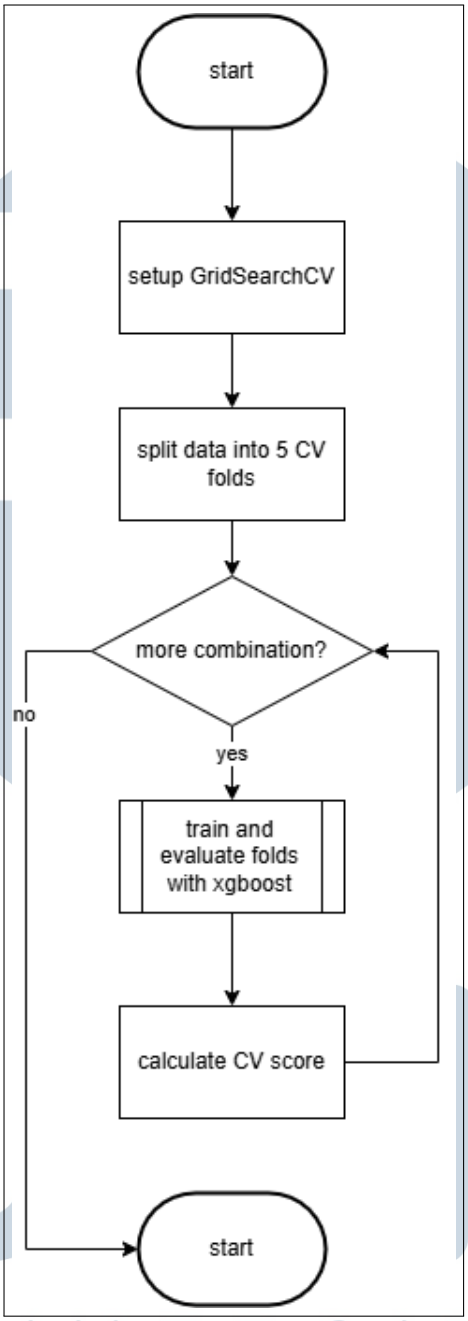
Selanjutnya dilakukan *label encoding*, yang berfungsi untuk mengubah label kategori menjadi representasi numerik (*index*) agar dapat diproses oleh XGBoost. *Label default* pada *dataset* asli menggunakan format numerik yang merepresentasikan kelas tumor, yaitu 1 untuk *meningioma*, 2 untuk *glioma*, dan 3 untuk *pituitary tumor*. Karena model XGBoost mengharuskan label dalam bentuk indeks berbasis nol (*zero-based indexing*), maka dilakukan proses *label encoding* untuk mengubah label menjadi 0 untuk *meningioma*, 1 untuk *glioma*, dan 2 untuk *pituitary tumor*. Setelah label dalam format indeks berbasis nol, tahap berikutnya adalah *setup hyperparameter grid*, hal ini berguna untuk menyiapkan kombinasi nilai-nilai *hyperparameter* yang akan diuji untuk menemukan konfigurasi atau kombinasi terbaik.

Tabel 3.3. *Grid hyperparameter* untuk model XGBoost

Hyperparameter	**Nilai yang Diuji**
`objective`	`multi:softmax`
`num_class`	3
`tree_method`	`hist`
`device`	`cuda`
`max_depth`	3, 6, 9
`min_child_weight`	3, 5
`learning_rate`	0.01, 0.1
`n_estimators`	100, 200, 300
`subsample`	0.8, 1.0
`colsample_bytree`	0.8, 1.0
`gamma`	0, 0.1
`reg_alpha`	0, 0.1
`reg_lambda`	0, 0.1

Tabel 3.3 menunjukkan jenis dan nilai *hyperparameter* yang digunakan untuk proses *tuning* model XGBoost. *Hyperparameter objective* dipilih ke *multi:softmax* karena tugas klasifikasi ini adalah multi-kelas, dengan jumlah kelas (*num_class*) sebanyak 3. Algoritma pohon menggunakan metode *histogram* (*tree_method=hist*) untuk efisiensi lalu seluruh proses dijalankan di GPU (*device=cuda*) untuk mempercepat komputasi. *Hyperparameter* seperti *max_depth, min_child_weight*, dan *learning_rate* digunakan untuk mengontrol kompleksitas dan performa model. *n_estimators* menentukan jumlah total pohon, sedangkan *subsample* dan *colsample_bytree* mengontrol jumlah sampel dan fitur yang digunakan pada tiap iterasi. *Hyperparameter gamma, reg_alpha*, dan *reg_lambda* adalah *hyperparameter* regularisasi yang berguna untuk menghindari *overfitting* pada model.

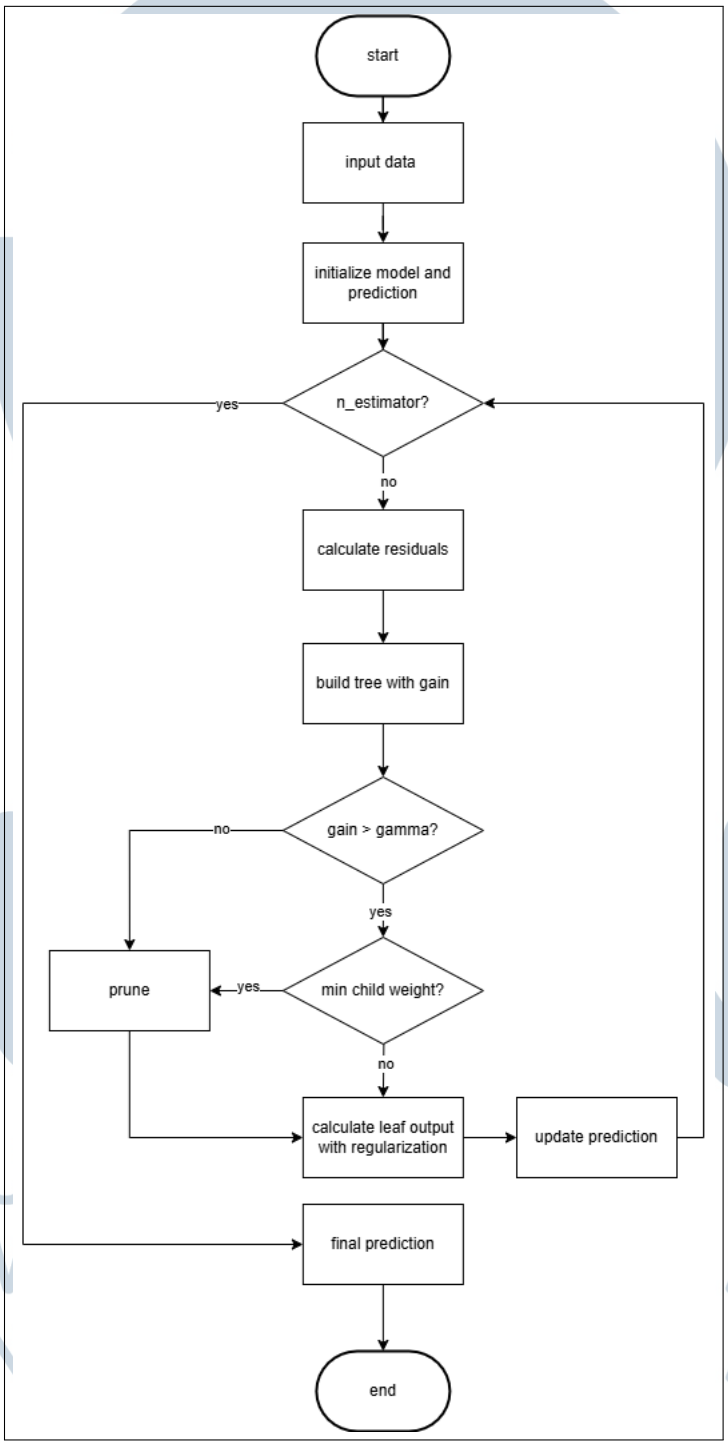
Kemudian dilakukan setup XGBoost, yaitu inisialisasi model XGBoost yang akan digunakan sebagai algoritma klasifikasi lalu proses pencarian dilakukan dengan *GridSearch*. Inisialisasi model menyangkut parameter *eval_metric* yaitu dengan *mlogloss* dan *early_stopping_rounds* yaitu 10. *GridSearch* adalah teknik pencarian sistematis terhadap semua kombinasi nilai *hyperparameter* yang telah disiapkan. *GridSearch* akan mengevaluasi model XGBoost dengan berbagai kombinasi *hyperparameter*.



Gambar 3.9. *Flowchart GridSearch*

Berdasarkan Gambar 3.9 tahapan dimulai dari melakukan *setup GridSearchCV* lalu dilakukan *data splitting* menjadi 5 bagian. Setelah itu, kombinasi *hyperparameter* akan dicoba satu-satu sampai ditemukan hasil yang paling maksimal. Dalam proses percobaan kombinasi *hyperparameter*, *GridSearchCV* akan melakukan *train* dan evaluasi terhadap data yang sudah di *split*.

Setelah proses pencarian dengan metode *GridSearch* selesai, langkah selanjutnya adalah *get best hyperparameter*, yaitu mengambil kombinasi *hyperparameter* terbaik yang memberikan akurasi paling optimal.



Gambar 3.10. *Flowchart train and evaluate folds with* XGBoost

Berdasarkan Gambar 3.10, tahapan dimulai dari melakukan *input* data lalu dilakukan inisialisasi model XGBoost dan lakukan prediksi naif/awal. Setelah itu *n_estimator* atau jumlah *tree* akan dicek, selama jumlah *tree* tidak melebihi batas yang diberikan maka proses XGBoost akan terus berjalan. Proses berlanjut ke tahapan penghitungan *residual* yaitu selisih antara prediksi dan nilai sebenarnya. Lalu XGBoost membangun *decision tree* dengan memilih split yang memiliki *gain* terbaik guna pemisahan data yang maksimal. *Split* yang memiliki *gain* tidak melebihi *threshold gamma* akan dipangkas (*pruning*). Selain itu, *split* juga harus memenuhi syarat *min_child_weight*. Setelah *tree* terbentuk, nilai output *leaf* akan dihitung dengan mempertimbangkan regularisasi. Prediksi model akan diperbarui dengan menambahkan hasil prediksi *tree* baru ke prediksi sebelumnya dengan mempertimbangkan *learning rate*. Proses ini berulang hingga jumlah *tree* mencapai *n_estimator*, lalu menghasilkan prediksi final yang merupakan akumulasi semua koreksi *tree*.

3.2.4 Model Evaluation

Setelah model XGBoost terbaik diperoleh melalui proses *Grid Search*, dilakukan evaluasi performa terhadap data evaluasi. Kemudian, performa model diukur menggunakan beberapa metrik evaluasi. Metrik yang digunakan adalah *accuracy*, *precision*, *recall*, dan F1-score. Untuk melihat distribusi kesalahan dan keberhasilan klasifikasi, digunakan *confusion matrix*, yang menggambarkan jumlah prediksi benar dan salah antar kelas.