# BAB III
# METODOLOGI PENELITIAN

## 3.1. Subjek dan Objek Penelitian

Subjek pada penelitian ini yaitu algoritma *Random Forest* pada klasifikasi risiko kredit. Sedangkan objek penelitiannya adalah *feature engineering* dan *hyperparameter tuning* model *Random Forest*.

## 3.2. Alat dan Bahan Penelitian

Alat dan bahan yang digunakan pada pengerjaan penelitian ini yaitu:

### 3.2.1. Perangkat Keras / *Hardware*

Perangkat keras yang digunakan pada penelitian ini adalah sebuah PC dengan spesifikasi sebagai berikut:

1. Intel Core i7-10700K 3.8Ghz Up To 5.1 Ghz Gygabyte Z490,
2. VGA GTX 1050 Ti 4Gb Memory 32Gb DDR4,
3. WDC 1TB SATA3 64MB,
4. ADATA SSD SU650 240 GB SATA III.

### 3.2.2. Perangkat Lunak / *Software*

Perangkat lunak yang digunakan pada penelitian ini sebagai berikut:

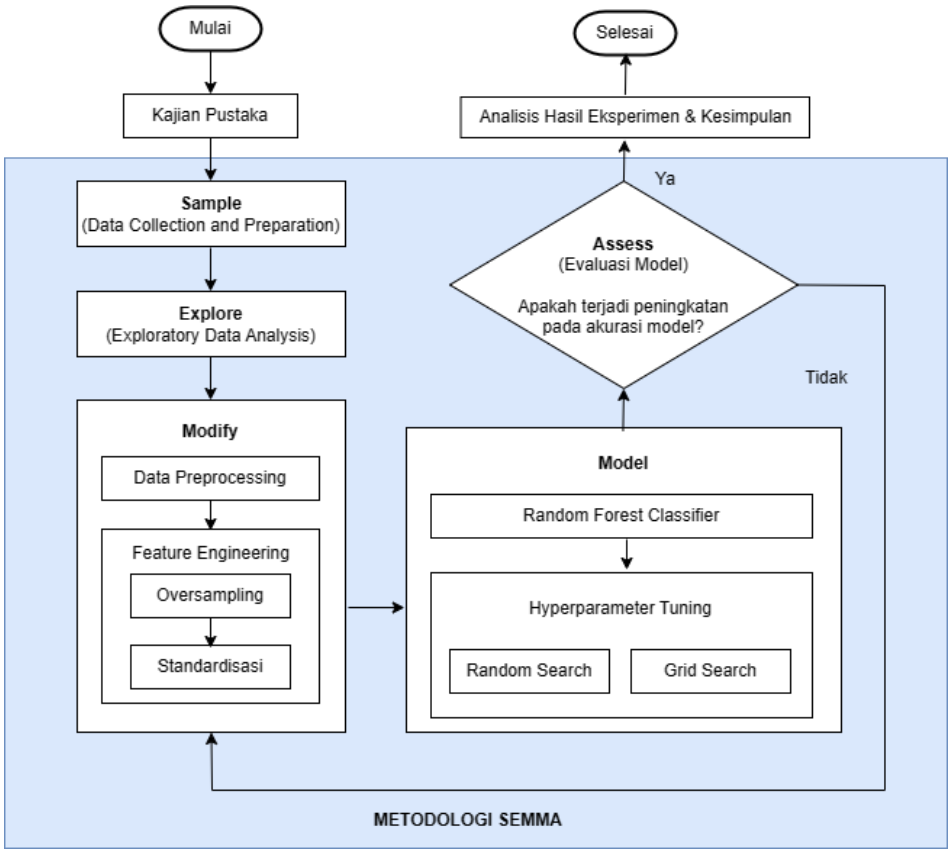
1. Sistem Operasi Windows 10
2. Jupyter Notebook (Anaconda)
3. Python 3.8
4. Microsoft Edge

### 3.2.3. Bahan / Data

Data yang digunakan pada penelitian ini berjudul “Credit Risk Data” yang didapatkan dari website *open source: kaggle.com*. Data berukuran 32.581 baris dengan 12 kolom [34].

## 3.3. Diagram Alir Penelitian

Dalam penelitian ini terdapat diagram alir yang akan menjelaskan proses penelitian seperti Gambar 3.1 berikut:




Gambar 3.1 Diagram Alir Penelitian

### 3.3.1. Kajian Pustaka

Pada alur pertama penelitian ini dilakukan eksplorasi mengenai kajian pustaka tentang algoritma *Random Forest*, *feature engineering*, *hyperparameter tuning* pada model *Random Forest*, risiko kredit, dan teknik evaluasi model *machine learning*, khususnya klasifikasi.

### 3.3.2. Metodologi SEMMA

Metodologi SEMMA merupakan salah satu pendekatan sistematis yang digunakan dalam *data mining* untuk mengelola proses analisis data.

#### 3.3.2.1. *Sample*

*Sample* merupakan tahapan pemilihan sampel data yang relevan, sampel data harus mewakili populasi dan mempertimbangkan tujuan analisis. Sampel yang digunakan pada penelitian ini berupa data risiko kredit yang didapatkan dari *kaggle* dengan judul "*credit risk data*" yang diunggah oleh Lao Tse. Fitur datanya terlihat pada Tabel 3.1.

Tabel 3.1 Fitur Data Risiko Kredit

| No. | Nama Fitur | Penjelasan |
|---|---|---|
| 1. | Person_age | Umur kreditur |
| 2. | Person_income | Pendapatan kreditur |
| 3. | Person_home_ownership | Kepemilikan rumah kreditur |
| 4. | Person_emp_length | Lama kerja kreditur |
| 5. | Loan_intent | Tujuan peminjaman |
| 6. | Loan_grade | Hasil pemeriksaan background |
| 7. | Loan_amnt | Jumlah pinjaman |
| 8. | Loan_int_rate | Suku bunga |
| 9. | Loan_percent_income | Presentase pinjaman |
| 10. | Cb_person_default_on_fire | Histori kegagalan |
| 11. | Cb_person_cred_hist_length | Jumlah histori peminjaman |
| 12. | Loan_status | Status kredit (Target) |

#### 3.3.2.2. *Explore*

*Explore* atau eksplorasi merupakan tahapan kedua pada metodologi SEMMA yang melibatkan eksplorasi data secara mendalam. Pada tahapan ini dilakukan proses *exploratory data analysis* (EDA) mengenai ukuran data, informasi fitur, statistik deskriptif, korelasi fitur, *missing value*, dan *outliers*. Namun sebelum melakukan *exploratory data analysis* (EDA), Dilakukan pengubahan nama kolom dari bahasa inggris ke bahasa indonesia agar lebih mudah dimengerti.

**Ukuran Data**

Ukuran data perlu diketahui pada saat melakukan *exploratory data analysis* (EDA). Salah satu cara melihat ukuran data yaitu menggunakan *library* pandas pada bahasa pemrograman *python*. Fungsi yang digunakan yaitu "*shape"*. Diketahui bahwa dataset terdiri dari 32581 baris dengan 12 kolom.

**Informasi Fitur**

Informasi fitur yang tertera meliputi nama fitur, jumlah baris, dan jenis fitur. Untuk melihat informasi fitur yang ada menggunakan fungsi "*info"* pada *pandas*.

Tabel 3.2 Informasi Fitur Data

| **Column** | **Non-Null Count** | **Dtype** |
|---|---|---|
| Umur | 32581 non-null | Int64 |
| Pendapatan | 32581 non-null | Int64 |
| KepemilikanRumah | 32581 non-null | Object |
| LamaKerja | 31686 non-null | Float64 |
| TujuanPeminjaman | 32581 non-null | Object |

| | | |
|---|---|---|
| HasilPemeriksaanBackground | 32581 non-null | Object |
| JumlahPinjaman | 32581 non-null | Int64 |
| SukuBunga | 29465 non-null | Float64 |
| %Pendapatan | 32581 non-null | Float64 |
| HistoryKegagalan | 32581 non-null | Object |
| JumlahHistoiPeminjaman | 32581 non-null | Int64 |
| StatusPinjaman (Var Target) | 32581 non-null | Int64 |

Pada Tabel 3.2 terlihat mengenai tipe data pada setiap fitur atau variabel. Tipe data yang ada berupa *integer*, *float*, dan *string*.

**Statistik Deskriptif**

Pada statistik deskriptif, dapat diketahui mengenai jumlah baris (*count*), *mean*, standar deviasi, Q1, Q2, Q3, nilai minimal, dan nilai maksimal seperti tertera pada Tabel 3.3.

Tabel 3.3 Hasil Statistik Deskriptif

| | **Count** | **Mean** | **Std** | **Min** | **25%** | **50%** | **75%** | **Max** |
|---|---|---|---|---|---|---|---|---|
| **Umur** | 32581 | 27.73 | 6.34 | 20 | 23 | 26 | 30 | 144 |
| **Pend** | 32581 | 66074 | 61983 | 4000 | 38500 | 55000 | 79200 | 6000000 |
| **LK** | 31686 | 4.78 | 4.14 | 0 | 2 | 4 | 7 | 123 |
| **JP** | 32581 | 9589 | 6322 | 500 | 5000 | 8000 | 12200 | 35000 |
| **SB** | 29465 | 11.01 | 3.24 | 5.42 | 7.90 | 10.99 | 13.47 | 23.22 |
| **%P** | 32581 | 0.17 | 0.1 | 0 | 0.09 | 0.15 | 0.23 | 0.83 |
| **JHP** | 32581 | 5.8 | 4.05 | 2 | 3 | 4 | 8 | 30 |
| **SP** | 32581 | 0.21 | 0.41 | 0 | 0 | 0 | 0 | 1 |

Pada Tabel 3.2 menunjukkan bahwa terdapat data yang tidak bersistribusi normal.

***Missing Value***

Data dikatakan *missing value* bila di baris dan kolom tersebut kosong/*Null*/*NaN*. Untuk melihat *missing value* menggunakan salah satu fungsi di *library* pandas yaitu *isnull*. Dataset yang terdeteksi kosong berjumlah 4011 records. 895 terdapat di kolom LamaKerja dan 3116 terdapat di kolom SukuBunga. Persentase data *Null* dipaparkan pada Tabel 3.4.

Tabel 3.4 Jumlah Data *Null*

| **Nama Kolom** | **Jumlah Data Null** | **Persentase (%)** |
|---|---|---|
| LamaKerja | 895 | 0.03% |
| SukuBunga | 3116 | 0.09% |

***Outliers***

Nilai pada data dikatakan *outliers* jika berada di bawah *lower bound* dan di atas *upper bound*. Berdasarkan persamaan 2.2 dan 2.3, maka jumlah data yang terdeteksi *outliers* terdapat pada Tabel 3.5.

Tabel 3.5 Jumlah Data *Outliers*

| **Nama Kolom** | **Jumlah Data Outliers** | **Persentase (%)** |
|---|---|---|
| Umur | 1494 | 0.046% |
| Pendapatan | 1484 | 0.045% |
| LamaKerja | 853 | 0.025% |
| JumlahPinjaman | 1689 | 0.051% |
| SukuBunga | 6 | 0.0001% |
| %Pendapatan | 651 | 0.019% |

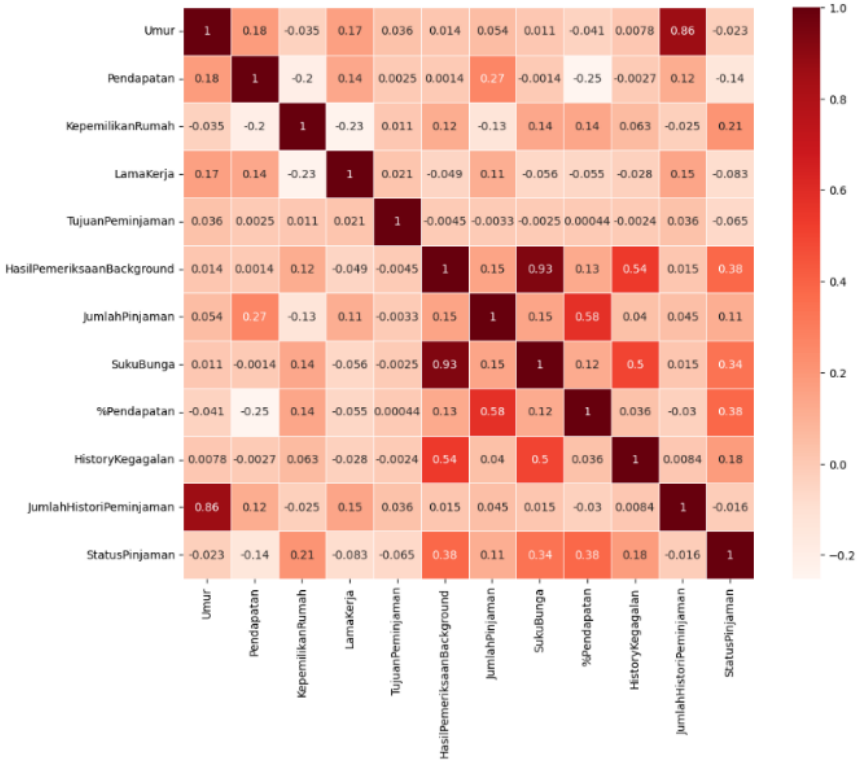
**Korelasi Fitur**

Untuk melihat korelasi fitur yang ada, digunakan rumus korelasi *pearson* seperti pada persamaan 2.4. Alur perhitungan korelasi *pearson* seperti pada Algoritma 3.1.

**Algoritma 3.1** Korelasi Fitur

Def fungsi *corr()* untuk objek DataFrame:
  Inisialisasi matriks korelasi ukuran (nxn), n adalah jumlah kolom DataFrame
  Untuk setiap pasangan kolom (i, j) dalam DataFrame:
    Hitung korelasi antara kolom i dan j menggunakan korelasi pearson
      IF korelasi ←pearson:
        Then $korelasi = \frac{n\Sigma xy - (\Sigma x)\,(\Sigma y)}{\sqrt{\{n\Sigma x^2 - (\Sigma x)^2\}\{n\Sigma y^2 - (\Sigma y)^2\}}}$
    Simpan nilai korelasi dalam matriks korelasi pada posisi (i, j) dan (j, i)
  Return matriks korelasi
Akhir Fungsi

Hasil nilai korelasi antar variabel menggunakan *pearson* pada dataset sesuai persamaan 2.4 dan sesuai dengan Algoritma 3.1 terdapat pada Gambar 3.2.




Gambar 3.2 Nilai Korelasi Antar Variabel

***Feature Importance***

Dalam *Random Forest*, *feature importance* mengukur seberapa penting setiap fitur terhadap kinerja keseluruhan model. Untuk mencari nilai *feature importance*, diperlukan *library scikit learn* dengan fungsi *ensemble* yang perhitungannya terdapat pada Algoritma 3.2.

**Algoritma 3.2** *Feature Importance*

Def fungsi hitung_nilai_entropi(data):

IF entropi ←*Entropy(S)*:

Then $Entropy(S) = \sum_{i=1}^{n} p_i \, log_2(p_i)$

Def fungsi hitung_information_gain(data, atribut):

IF information_gain ←*Gain(A)*:

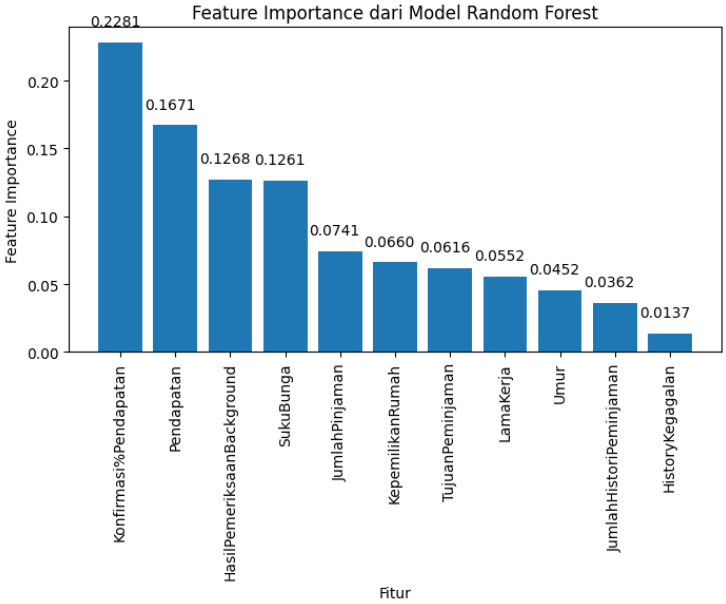
Then $Gain(A) = Entropy(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} \times Entropy(S_i)$

Def fungsi cari atribut_pemisah_terbaik(data, atribut_kandidat):

Gain_max = - infinity

```
    Atribut_terbaik = null
    Untuk setiap atribut in atribut_kandidat:
        Gain_i ←hitung_information_gain(data, atribut)
        IF Gain_i > Gain_max:
            Gain_max ← Gain_i
            atribut_terbaik ←atribut
    Return atribut_terbaik
Def fungsi hitung feature_importance(data, all_atribut):
    feature_importance ←{}
    Untuk setiap atribut in all_atribut:
        Data_cln ←clone(data) – untuk menghindari perubahan data asli
        Data_cln.del_atribut(atribut) – menghilangkan atribut yang dievaluasi
        DTree ←bangun_pohon_keputusan(Data_cln, atribut_kandidat)
        Importance ←hitung_information_gain(data, atribut)
        feature_importance[atribut] ← Importance
    Return feature_importance
Akhir Fungsi
```

Nilai *feature importance* yang dihitung menggunakan Algoritma 3.2 terdapat pada Gambar 3.3.




Gambar 3.3 Nilai *Feature Importance*

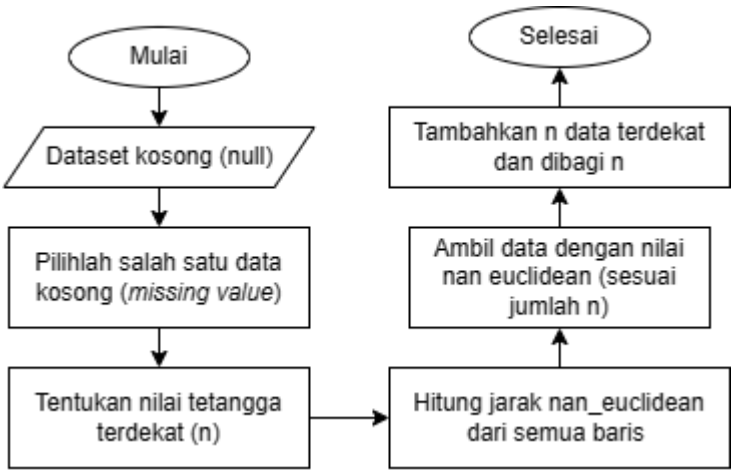
Dari Gambar 3.3 terlihat bahwa nilai yang didapat masih di bawah 0.5 namun masih di atas 0 dengan nilai *feature importance* tertinggi yaitu fitur Konfirmasi%Pendapatan yaitu 0.2281, artinya fitur tersebut adalah fitur yang paling berpengaruh terhadap kebaikan model *Random Forest*.

**3.3.2.3.*Modify***

*Modify* atau modifikasi merupakan tahapan ketiga pada metodologi SEMMA yang melibatkan teknik – teknik pemrosesan data. Pada tahapan ini akan dilakukan *data preprocessing* yang meliputi *handle missing value, handle outliers, replace value, encoding data,* dan *feature engineering* yang meliputi *oversampling* menggunakan *random oversampling* dan standardisasi data menggunakan *Robust Scaler*.

***Handling Missing Value***

*Handling missing value* merupakan satu diantara *data preprocessing* yang berfokus pada nilai kosong atau *Null*. *Handling missing value* menggunakan *KNNImputer*. *KNNImputer* adalah metode imputasi data yang menggunakan algoritma *K-Nearest Neighbors* (KNN) untuk mengisi nilai-nilai yang hilang dalam sebuah dataset berdasarkan nilai-nilai tetangga terdekatnya. Alur kerja *KNNImputer* terdapat pada Gambar 3.3.




Gambar 3.4 Alur Kerja *KNNImputer*

Dari Gambar 3.4 dapat diartikan bahwa cara mengatasi data kosong dengan nilai tetangga terdekat dimualai dari pengambilan salah satu nilai pada data yang kosong lalu menentukan nilai n (tetangga terdekat). Setelah itu dihitung nilai *nan*

*euclidean* dari semua baris sesuai jumlah n. Setelah itu n *euclidean* terkecil akan ditambahkan dan dibagi dengan nilai n. Berikut contoh perhitungan *KNNImputer* pada data kosong. Tabel 3.6 menunjukkan contoh data yang terdapat *missing value*.

Tabel 3.6 Contoh Perhitungan *KNNImputer*

| | Umur | KR | LK | TP | HPB |
|---|---|---|---|---|---|
| 0 | 21 | 3 | 5 | 2 | 2 |
| 1 | 25 | 1 | 1 | 4 | 3 |
| 2 | Null | 4 | 4 | 4 | 3 |
| 3 | 24 | 4 | 8 | 4 | 3 |
| 4 | 21 | 3 | 2 | 6 | 1 |

$distance1 = \frac{4}{4}\ x\ ((4-1)^2 + (4-1)^2 + (4-4)^2 + (3-3)^2)½ = 4242$

$distance2 = \frac{4}{4}\ x\ ((4-3)^2 + (4-5)^2 + (4-2)^2 + (3-2)^2)½ = 2645$

$$distance3 = \frac{4}{4}\ x\ ((4-3)^2 + (4-2)^2 + (4-6)^2 + (3-1)^2)½ = 3605$$

$$distance4 = \frac{4}{4}\ x\ ((4-4)^2 + (4-8)^2 + (4-4)^2 + (3-3)^2)½ = 4$$

Diketahui nilai *euclidean* terkecil yaitu *distance* 2 dan *distance* 3 (2.645 dan 3.605) maka data yang kosong akan diinputkan dari tetangga *distance* 2 dan 3. Jadi nilai hasil *KNNImputer* untuk mengisi data kosong yaitu:

$$\text{Data Inputan} = \frac{21 + 21}{2}$$

$$\text{Data Inputan} = 21$$

***Replace Value***

Data pada kolom "Hasil Pemeriksaan Background" bersifat kategorik namun memiliki urutan (*ordinal dataset*) sehingga tidak dapat diubah ke numerik secara acak menggunakan teknik *encode*. Data yang sifatnya ordinal akan diubah secara manual dengan fungsi *'replace'* pada bahasa pemrograman python.

Selain itu terdapat data pada kolom "%Pendapatan" yang nilainya berbeda dengan hasil perhitungan secara manual. Untuk itu akan dilakukan perhitungan ulang menggunakan persamaan 3.1 dan sesuai dengan algoritma 3.3.

$$\text{Rumus \%Pendapatan} = \frac{\text{data}['\text{JumlahPinjaman}']}{\text{data}['\text{Pendapatan}']} \qquad (3.1)$$

**Algoritma 3.3** Perhitungan %Pendapatan

```
Def fungsi hitung_persentase_pendapatan(data):
    JumlahPinjaman ← data['JumlahPinjaman']
    Pendapatan ← data['Pendapatan']
    Jika Pendapatan tidak sama dengan 0:
        persentase_pendapatan ← (JumlahPinjaman / Pendapatan) * 100
    Lainnya:
        persentase_pendapatan ← 0
    Menampilkan persentase_pendapatan
Akhir Fungsi
```

***Encoding***

*Encoding* data merupakan perubahan data teks ke numerik sehingga data dapat dikenali oleh mesin. *Encoding data* dilakukan menggunakan *Label Encoder* agar tidak menambah dimensi fitur pada data. Tabel 3.7 merupakan contoh perubahan setelah dilakukan *Label Encoder*.

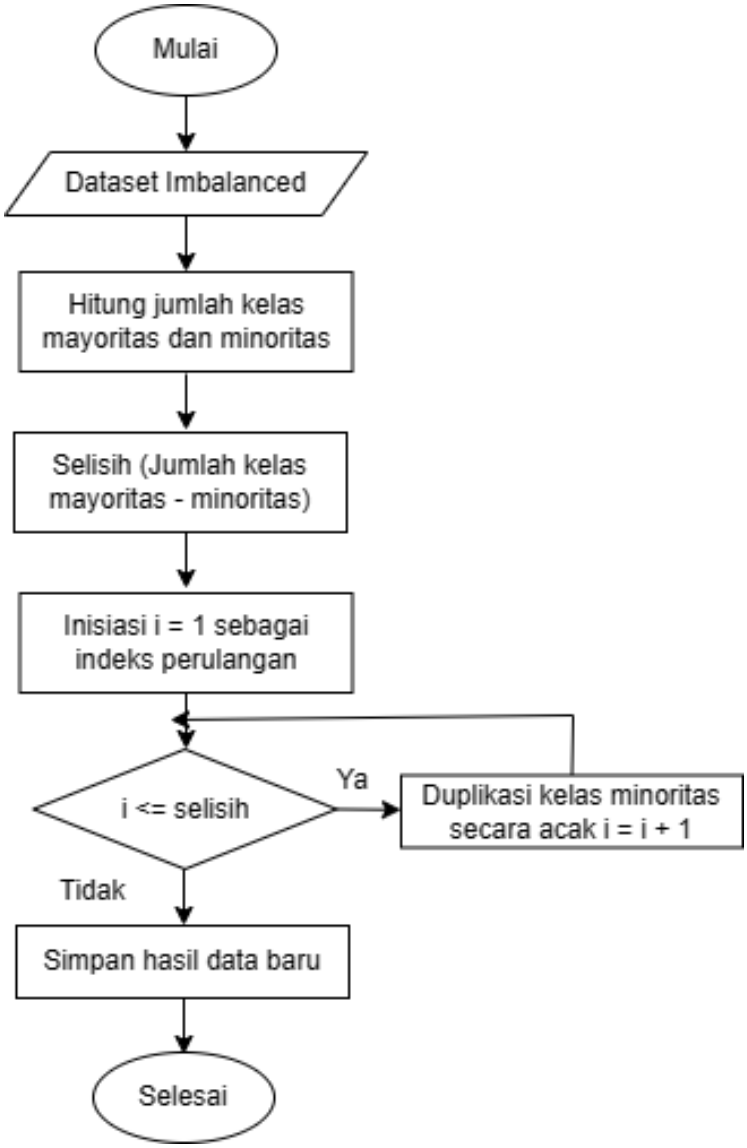
Tabel 3.7 Contoh Penerapan *Label Encoder*

| Sebelum Encode Data | Setelah Encode Data |
|---|---|
| Education | 0 |
| Medical | 1 |
| Personal | 2 |

Dari Tabel 3.7 terlihat bahwa jenis data sebelum dilakukan *encode* yaitu teks, sedangkan setelah dilakukan *encoding data* maka data tersebut akan menjadi numerik.

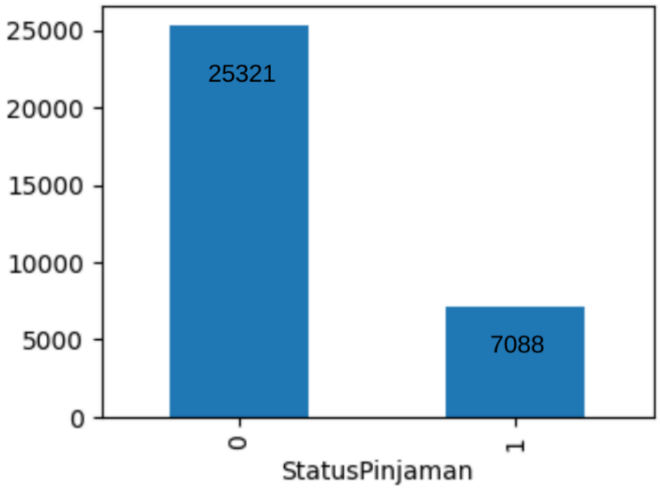
***Oversampling***

Tahap selanjutnya yaitu *sampling* data yang menerapkan *oversampling* sehingga data akan bertambah. *Oversampling* menggunakan teknik *random oversampling.* Alur kerja *random oversampling* terdapat pada Gambar 3.5.



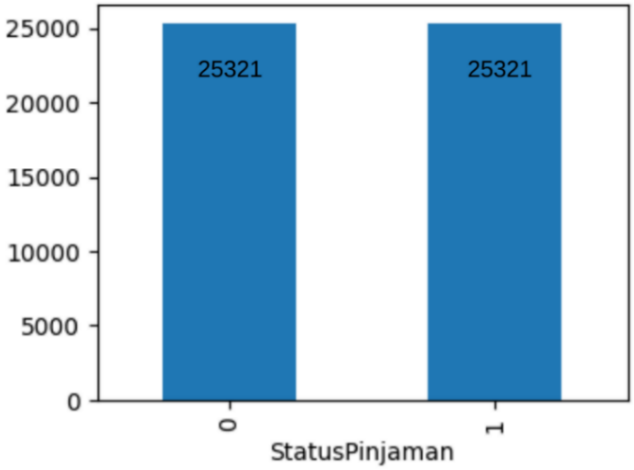

Gambar 3.5 Alur Kerja *Random Oversampling*

Cara kerja *random oversampling* dimulai ketika dataset terdeteksi *imbalanced,* kemudian menghitung jumlah kelas mayoritas dan minoritas. Setelah itu diketahui selisih antara kelas mayoritas dan minoritas. Dilanjutkan dengan inisiasi i = 1 sebagai indeks perulangan. Jika i <= nilai selisih maka akan dilakukan duplikasi kelas minoritas secara acak dengan ditambahkan 1. Namun jika i sudah mencukupi maka disimpanlah data baru dan selesai.




Gambar 3.6 Contoh Perbandingan Kelas Pada Data

Gambar 3.6 menunjukkan kelas data yang tidak seimbang (*imbalanced*) sehingga perlu dilakukan penyerataan kelas dengan menambahkan data pada kelas minoritas (1) atau *oversampling*. Jumlah data kelas 1 yaitu 25321 baris, sedangkan kelas 0 berjumlah 7088. Setelah dilakukan *oversampling*, data berjumlah 50.642 baris yang terlihat pada Gambar 3.7.




Gambar 3.7 Contoh Data Hasil *Oversampling*

**Standardisasi**

Standardisasi data dilakukan untuk menyeragamkan skala data. Berikut merupakan contoh perhitungan standardisasi menggunakan *Robust Scaler* yang sesuai dengan persamaan 2.5.

x’ = (9600 – 79478) / (95000 – 11291.75)

x’ = -0.83

Contoh data sebelum dilakukan standardisasi yang terdapat pada Gambar 3.8.

| Umur | Pendapati | Kepemilik | LamaKerja | HasilPeme | JumlahPir | SukuBung | %Pendapa | HistoryKegagal | JumlahHis | StatusPinjaman |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 9600 | 3 | 5 | 2 | 1000 | 11.14 | 0.1 | 1 | 2 | 0 |
| 23 | 115000 | 4 | 2 | 1 | 35000 | 7.9 | 0.3 | 1 | 4 | 0 |
| 23 | 500000 | 1 | 7 | 2 | 30000 | 10.65 | 0.06 | 1 | 3 | 0 |
| 23 | 120000 | 4 | 0 | 1 | 35000 | 7.9 | 0.29 | 1 | 4 | 0 |
| 25 | 162500 | 4 | 2 | 1 | 35000 | 7.49 | 0.22 | 1 | 4 | 0 |
| 25 | 137000 | 4 | 9 | 5 | 34800 | 16.77 | 0.25 | 2 | 2 | 0 |
| 24 | 10980 | 3 | 0 | 1 | 1500 | 7.29 | 0.14 | 1 | 3 | 0 |
| 22 | 59000 | 4 | 123 | 4 | 35000 | 16.02 | 0.59 | 2 | 3 | 1 |
| 25 | 9600 | 1 | 1 | 3 | 5500 | 12.87 | 0.57 | 1 | 3 | 1 |
| 23 | 65500 | 4 | 4 | 3 | 35000 | 15.23 | 0.53 | 1 | 2 | 1 |

Gambar 3.8 Data Sebelum Standardisasi

Contoh data hasil standardisasi menggunakan *Robust Scaler* pada kolom Umur, Pendapatan, LamaKerja, JumlahPinjaman, dan SukuBunga terdapat pada Gambar 3.9.

| Umur | Pendapatan | Kepemilik | LamaKerja | HasilPeme | JumlahPinjama | SukuBunga | %Pendapa | HistoryKe | JumlahHis | StatusPinja |
|---|---|---|---|---|---|---|---|---|---|---|
| -1 | -0.8305437 | 3 | 0.25 | 2 | -1.176579581 | 0.121070683 | 0.1 | 1 | 2 | 0 |
| 0 | 0.42220116 | 4 | -0.5 | 1 | 0.083379655 | -0.432612842 | 0.3 | 1 | 4 | 0 |
| 0 | 4.99816664 | 1 | 0.75 | 2 | -0.101908468 | 0.037334594 | 0.06 | 1 | 3 | 0 |
| 0 | 0.48162928 | 4 | -1 | 1 | 0.083379655 | -0.432612842 | 0.29 | 1 | 4 | 0 |
| 1 | 0.98676833 | 4 | -0.5 | 1 | 0.083379655 | -0.502677732 | 0.22 | 1 | 4 | 0 |
| 1 | 0.6836849 | 4 | 1.25 | 5 | 0.07596813 | 1.083181252 | 0.25 | 2 | 2 | 0 |
| 0.5 | -0.8141415 | 3 | -1 | 1 | -1.158050769 | -0.536855728 | 0.14 | 1 | 3 | 0 |
| -0.5 | -0.2433938 | 4 | 29.75 | 4 | 0.083379655 | 0.95501377 | 0.59 | 2 | 3 | 1 |
| 1 | -0.8305437 | 1 | -0.75 | 3 | -1.009820271 | 0.416710343 | 0.57 | 1 | 3 | 1 |
| 0 | -0.1661373 | 4 | 0 | 3 | 0.083379655 | 0.820010688 | 0.53 | 1 | 2 | 1 |

Gambar 3.9 Data Setelah Standardisasi

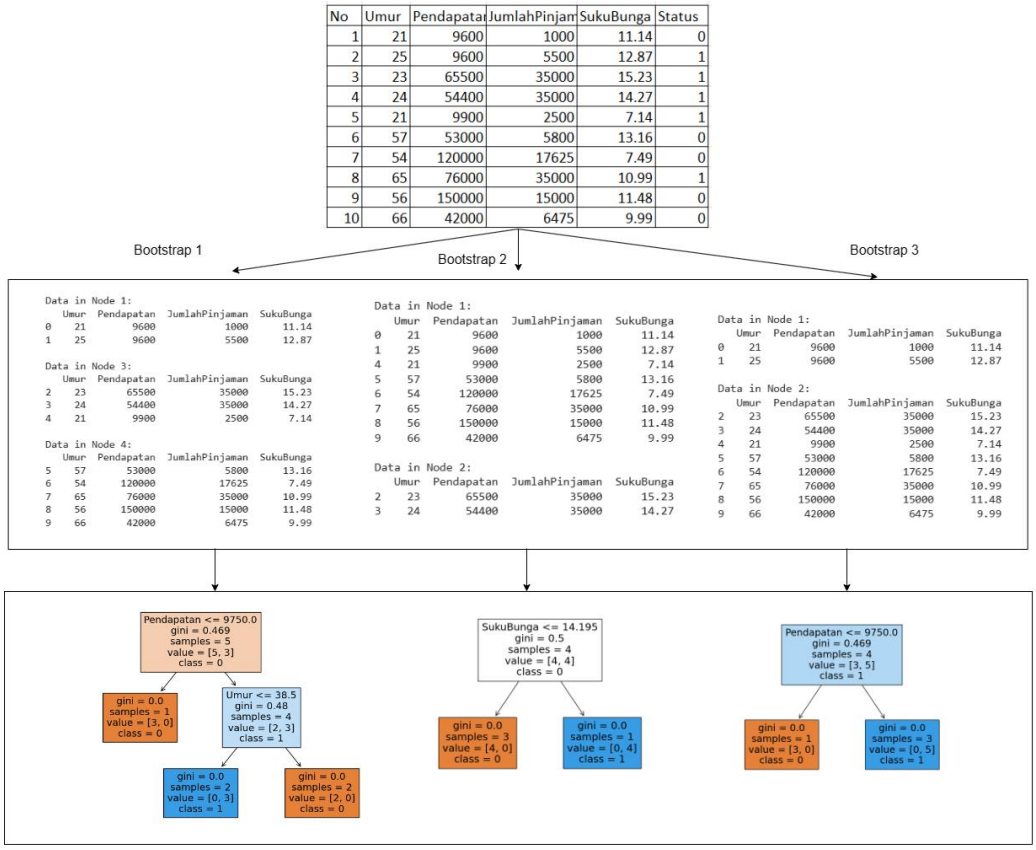
### 3.3.2.4.Model

Model merupakan tahapan keempat pada metodologi SEMMA yang melibatkan pembuatan model algoritma *machine learning*. Pada tahapan ini akan dilakukan pemodelan menggunakan algoritma *Random Forest* dan dilakukan *hyperparameter tuning* untuk mencari parameter paling optimal.

Langkah – langkah cara kerja algoritma *Random Forest* terdiri dari:

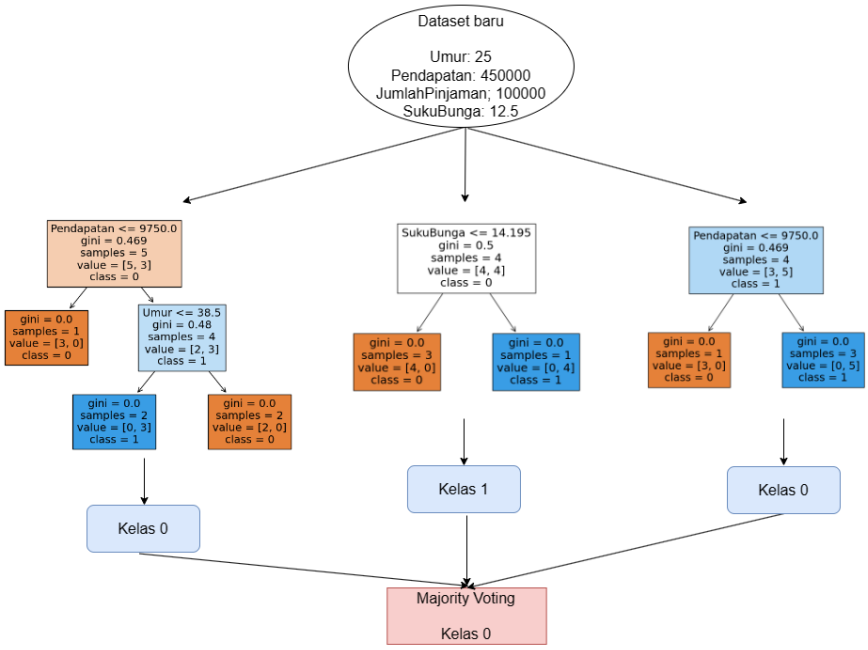
1. Penentuan parameter *Random Forest* (jumlah pohon (*bootstrap*), kedalaman pohon maksimal, jumlah maksimal fitur, kriteria (*gini* atau *entropy*), jumlah minimal sampel agar node bisa terbagi, dan jumlah minimal sampel agar node menjadi daun).
2. Pembuatan *bootstrap* – *bootstrap* dari data.
3. Pembuatan pohon dari masing – masing *bootstrap*.
4. Prediksi kelas data baru berdasarkan pohon – pohon yang terbentuk.
5. Menentukan kelas data berdasarkan *majority voting*.

Langkah - langkah tersebut dijelaskan pada Gambar 3.10 dan 3.11.




Gambar 3.10 Algoritma *Random Forest*

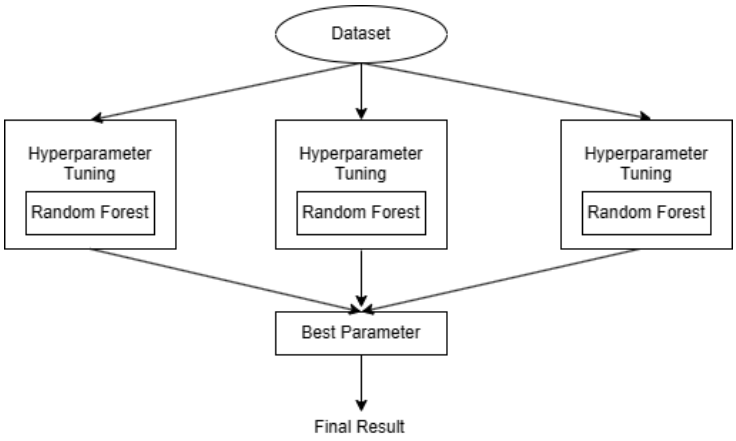
Gambar 3.10 menjelaskan cara kerja algoritma *Random Forest* untuk data risiko kredit. Algoritma *Random Forest* terdiri dari beberapa *bootstrap* sehingga dari kumpulan data akan membagi secara acak ke dalam *bootstrap* - *bootstrap* yang berbeda sesuai dengan parameter - parameter yang ditentukan.




Gambar 3.11 Contoh Penggunaan *Random Forest* Pada Data Baru

Pada Gambar 3.11, dengan menginputkan data baru maka *Random Forest* akan memprediksi bahwa data tersebut masuk ke kelas 0 berdasarkan *voting* dari 3 pohon atau *bootstrap*.

Parameter yang akan dilakukan dieksplorasi meliputi jumlah pohon (*n_estimators*), kriteria (*criterion*), maksimal kedalaman pohon (*max_depth*), jumlah fitur yang dipertimbangkan (*max features*), jumlah sampel minimum untuk pembentukan *node* baru (*min_samples_split*), dan jumlah sampel minimum pada daun (*min_samples_leaf*). Penelitian ini menginputkan beberapa nilai pada setiap parameter model *Random Forest* dan dicari hasil akurasi terbaik.




Gambar 3.12 Penerapan *Hyperparameter Tuning* Pada *Random Forest*

Gambar 3.12 menunjukkan penggunaan *hyperparameter tuning* pada model *Random Forest* untuk mendapatkan parameter terbaik. Untuk mencari parameter terbaik tersebut, akan menggunakan *Grid Search CV* dan *Randomized Search CV* yang merupakan teknik optimasi dengan bantuan *library Scikit Learn*.

#### 3.3.2.5. *Assess*

*Assess* atau penilaian merupakan tahapan terakhir pada metodologi SEMMA yang melibatkan penilaian atau *evaluation*. Pada tahapan ini akan mengevaluasi model *Random Forest* menggunakan *Accuracy, Confusion Matrix, Classification Report*, dan nilai AUC. *Accuracy* mengukur kebaikan model dalam memprediksi jawaban benar dan merupakan rasio antara jawaban benar dari total jawaban. *Confusion Matrix* menilai performa model dengan memperhatikan jumlah jawaban benar dan salah yang dilihat dari nilai *True Positive, True Negative, False Positive,* dan *False Negative. Classification Report* mengevaluasi nilai *preciision, recall,* dan *F1-Score*. Nilai AUC mengukur kebaikan model dalam membedakan antara kelas positif dan negatif. Nilai AUC merupakan area di bawah kurva ROC.

### 3.3.3. Analisis Hasil Eksperimen dan Kesimpulan

Pada tahapan ini, akan dilakukan analisis hasil akurasi model *Random Forest* dengan 3 perlakuan, seperti pada Tabel 3.8.

Tabel 3.8 Perlakuan Terhadap Pemodelan

| | Model 1 | Model 2 | Model 3 | Model 4 |
|---|---|---|---|---|
| *Feature Engineering* (*Oversampling & Standardisasi*) | Tidak | Ya | Ya | Ya |
| *Hyperparameter Tuning* (*Random Search*) | Tidak | Tidak | Ya | Tidak |
| *Hyperparameter Tuning* (*Grid Search*) | Tidak | Tidak | Tidak | Ya |

Dengan 4 model yang akan dibuat, peneliti dapat membandingkan model terbaik dan dapat mengetahui teknik *feature engineering* serta *hyperparameter tuning* terbaik yang dapat diterapkan pada data risiko kredit, khususnya "*credit risk data"* oleh Lao Tse [31].