Implementasi Algoritma *Decision Tree* untuk Klasifikasi Pelanggan Aktif atau Tidak Aktif pada Data Bank

Rafael Nuansa Ramadhon[1], Aldino Ogi[2], Ari Permana Agung[3],
Ryandra Putra[4], Siti Sarah Febrihartina[5], Uus Firdaus[6]

[1]Ilmu Komputer, Universitas Djuanda, Indonesia
[2]Ilmu Komputer, Universitas Djuanda, Indonesia
[3]Ilmu Komputer, Universitas Djuanda, Indonesia
[4]Ilmu Komputer, Universitas Djuanda, Indonesia
[5]Ilmu Komputer, Universitas Djuanda, Indonesia
[1]rafael.nuansa.ramadhon@unida.ac.id

ABSTRAK

Penelitian ini mengeksplorasi penerapan pohon keputusan (*decision tree*) sebagai metode analisis dalam konteks pengambilan keputusan. *Decision tree* merupakan algoritma *machine learning* yang populer karena kemampuannya dalam memodelkan dan memahami hubungan kompleks di antara variabel-variabel yang terlibat dalam suatu keputusan. Dalam penelitian ini, kami berfokus pada penerapan algoritma *decision tree* untuk klasifikasi pelanggan aktif dan tidak aktif di sektor perbankan. Klasifikasi dalam *machine learning* adalah proses membangun model atau algoritma yang mampu mengidentifikasi dan mengelompokkan data ke dalam kategori atau kelas tertentu. Tujuan utama dalam penelitian ini yaitu untuk mengidentifikasi faktor-faktor yang paling berpengaruh dalam menentukan status pelanggan, terutama dalam pemasaran layanan perbankan. Kami menggunakan *dataset* yang mencakup berbagai atribut diantaranya data lokasi, ID produk, nama pelanggan, saldo, dan status.

Kata Kunci: *Machine Learning*, *Decision Tree*, Klasifikasi

PENDAHULUAN

Di era digital ini, industri perbankan menghadapi tantangan besar dalam mengelola dan memahami informasi yang semakin berkembang. Di perusahaan mana pun termasuk lembaga keuangan, data adalah aset berharga yang dapat digunakan untuk strategi perusahaan (Firdaus & Utama, 2020). Data pelanggan menjadi aset berharga yang dapat memberikan wawasan mendalam tentang perilaku konsumen dan membantu bank (khususnya bagian pemasaran) untuk proses pengambilan keputusan yang cerdas. Oleh karena itu, teknik data mining, khususnya penggunaan algoritma *Decision Tree*, dianggap sebagai pendekatan yang efektif untuk mengklasifikasikan dan memahami pola perilaku pelanggan.

Keputusan yang diambil oleh pelanggan di sektor perbankan dipengaruhi oleh berbagai faktor, termasuk riwayat transaksi, preferensi layanan, dan karakteristik keuangan. Algoritma *Decision Tree* telah terbukti efektif dalam mengidentifikasi pola dan hubungan kompleks dalam data, menjadikannya pilihan yang tepat untuk analisa klasifikasi pelanggan. Melalui penelitian ini, diharapkan dapat ditemukan pola perilaku pelanggan yang belum teridentifikasi sebelumnya, yaitu status aktif atau tidak aktif. Penelitian ini juga dapat digunakan untuk mengoptimalkan strategi pemasaran, meningkatkan retensi pelanggan, dan mengurangi risiko keuangan. Selain itu, penerapan algoritma *Decision Tree* di lingkungan perbankan dapat memberikan kontribusi signifikan pada literatur analisis data di sektor ini.

METODE PENELITIAN

A. Machine Learning

Machine Learning adalah suatu cabang dari kecerdasan buatan (*Artificial Intelligence* atau *AI*) yang fokus pada pengembangan sistem yang mampu belajar dan meningkatkan kinerjanya dari pengalaman atau data tanpa perlu pemrograman yang eksplisit. Dengan kata lain, *machine learning* dapat mengidentifikasi pola dari data dan menggunakan pola-pola tersebut untuk membuat keputusan atau melakukan tugas tanpa perlu diprogram secara khusus.

Pada konteks *machine learning*, terdapat berbagai algoritma dan teknik yang digunakan untuk mengolah dan memahami data, membangun model, serta membuat prediksi atau keputusan. *Machine Learning* dapat diterapkan dalam berbagai bidang seperti pengenalan wajah, pengenalan suara, analisis data, pengenalan pola, dan masih banyak lagi.

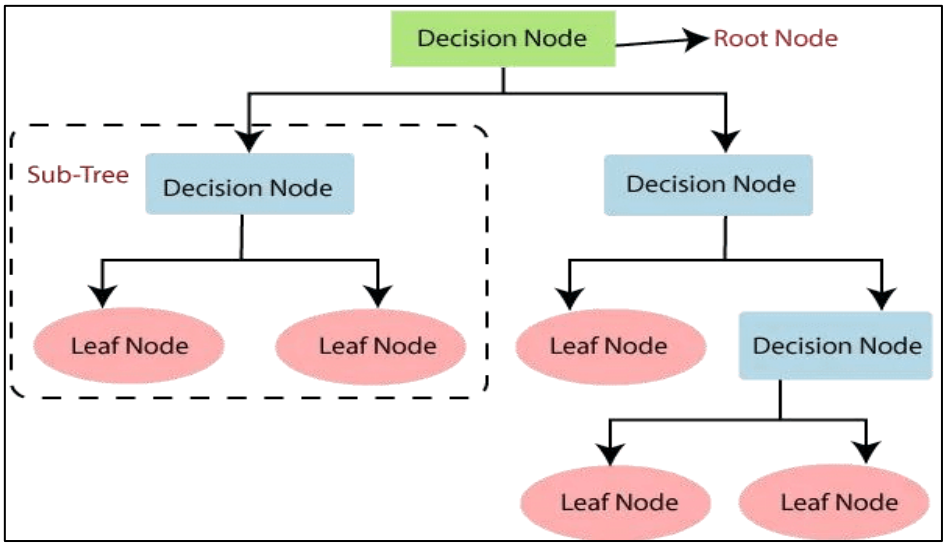
B. Algoritma Decision Tree

Decision Tree merupakan struktur pohon yang terdiri dari node-node yang merepresentasikan keputusan dan cabang-cabang yang merepresentasikan konsekuensi dari sebuah keputusan (Feby, 2023). *Decision Tree* merupakan model prediksi yang bersifat *supervised* yang berarti memerlukan *training dataset* yang perannya menggantikan pengalaman manusia di masa lalu dalam membuat keputusan (Kurniawan, 2020).

Cara untuk membuat model *Decision Tree* adalah dengan memecah data ke dalam kelompok yang lebih kecil berdasarkan atribut di dalam data. Pembagian kelompok ini dilakukan berulang kali hingga seluruh elemen data yang berasal dari kelas yang sama dapat masuk ke dalam satu kelompok (Kurniawan, 2020). Terdapat beberapa algoritma yang umum digunakan untuk melatih model *Decision Tree*, salahsatunya bernama CART (*Classification and Regression Trees*)

yang diterapkan dalam Scikit-learn. CART merupakan turunan dari algoritma lain bernama C4.5 yang merupakan turunan dari algoritma ID3. Ross Quilan (seorang peneliti bidang *Machine Learning*) membuat sebuah algoritma yang dikenal dengan ID3 (*Iterative Dichotomiser*) tersebut (Puspitorini, 2021).

Dalam *Decision Tree* terdapat istilah-istilah pada rangkaian pohon yang dibuat seperti *Decision Node* (*feature* yang digunakan untuk membuat keputusan), *decision node* yang paling atas dinamakan *root node*. Lalu ada *Leaf Node* (output dari keputusan) dan *Sub Tree* yaitu bagian atau cabang dari pohon. Agar lebih jelas dapat kita lihat visualisasi *decision tree* pada Gambar 1 (Wadhwatanya, 2021).



Gambar 1 Konsep Decision Tree

Pada dasarnya, algoritma ID3 melakukan *splitting* (pemecahan) data ke dalam dua kelompok berdasarkan atribut-atribut yang ada di dalam data, dengan mengukur suatu angka yang disebut ***entropy***. *Entropy* yaitu sebagai ukuran seberapa acak suatu kelompok data. Proses *splitting* dilakukan bertingkat dari atas ke bawah. Rumus matematika *Entropy* dapat dilihat pada Gambar 2.

$$\text{Entropy(S)} = \text{-}\ Entropy = -\sum_{i=1}^{n} pi * log_2\,(pi)$$

Gambar 2 Rumus Matematika Entropy

Keterangan :
S : Himpunan kasus
n : Jumlah partisi s
pi : Jumlah kasus pada partisi ke-i

Setelah pemilihan atribut, langkah berikutnya adalah pemrosesan melalui *machine learning* algoritma *decision tree*. *Decision tree* ini menentukan akar teratas dengan memulai dari atribut yang memiliki bobot tertinggi, dilanjutkan dengan proses pemisahan (*splitting process*).

Rumus yang digunakan dalam perhitungan *gain* setelah melakukan perhitungan *entropy* dengan n adalah jumlah kelas, dan p adalah proporsi nilainilai masuk ke dalam kelas di tingkat i.

Gain yaitu informasi yang diperoleh dari modifikasi *entropy* pada sebuah kumpulan banyaknya data , baik melalui pemeriksaan atau dengan kata lain ringkas dengan cara melakukan partisipasi terhadap suatu set data. Pada sebuah *entropy* memiliki atribut untuk menjadikan *entropy* tersebut sebagai porosnya. Secara ilmiah pada sebuah atribut cabang pohon dari algoritma ini diperbesar sehingga seluruh pohon terbentuk. Pada sebuah atribut yang ada tidak semuanya digunakan karena mengalami seleksi atribut. Proses ini memanfaatkan tingkat informasi (*information gain*). Tingkat informasi dalam pohon keputusan adalah suatu proses untuk memberikan nilai bobot pada setiap atribut, sehingga tidak semua atribut yang tersedia akan dimasukkan ke dalam proses *machine learning* pada pohon keputusan. Rumus Matematika Gain dapat dilihat pada Gambar 3:

$$Gain(S,A) = Enthropy(S) - \sum_{i=1}^{n} - \frac{|Si|}{|S|} * Enthopy(Si)$$

Gambar 3 Rumus Matematika Gain

Keterangan :

S : Himpunan kasus

n : Jumlah partisi atribut

A|Si| : Jumlah kasus pada partisi ke -i

|S| : Jumlah kasus dalam S

Dengan memahami formula-formula di atas, data yang telah diperoleh dapat dimasukkan dan diolah menggunakan algoritma tersebut untuk proses pembuatan pohon keputusan atau *decision tree*

Keuntungan dari algoritma *decision tree* meliputi kemampuan interpretasi, kemampuan menangani data kategori dan numerik, serta kecenderungan untuk tidak memerlukan normalisasi atau standarisasi data. Namun, mereka dapat rentan terhadap *overfitting*, di mana model terlalu sesuai dengan data pelatihan dan mungkin tidak dapat melakukan generalisasi dengan baik pada data baru.

Dari pohon keputusan, keunggulannya terletak pada kemampuannya menghasilkan model yang relatif mudah diinterpretasikan. Secara spesifik, pohon ini dapat diartikan sebagai kumpulan aturan keputusan, di mana setiap aturan terdiri dari keputusan di node internal sepanjang jalur ke simpul daun, dan hasilnya menjadi label simpul daun. Selain itu, karena area-areanya saling terpisah dan meliputi seluruh ruang, kumpulan aturan ini dapat dianggap sebagai opsi alternatif atau disjungsi

Secara umum algoritma C4.5 untuk membangun pohon keputusan adalah sebagai berikut :

1. Menetapkan atribut sebagai akar.
2. Membangun cabang untuk setiap nilai.
3. Membagi kasus di dalam setiap cabang.
4. Verifikasi dan ulangi proses dengan tujuan setiap cabang hingga semua kasus pada cabang memiliki kelas yang serupa.

1) **Keunggulan Metode Pohon Keputusan**

Kelebihan dari pendekatan pohon keputusan melibatkan:

- Simpulan daerah pengambilan keputusan yang sebelumnya kompleks dan sangat global, dapat disederhanakan menjadi lebih ringkas dan spesifik.
- Penghapusan perhitungan yang tidak perlu, karena saat menggunakan metode pohon keputusan, sampel diuji hanya berdasarkan kriteria atau kelas tertentu.
- Fleksibilitas untuk memilih fitur dari node internal yang berbeda, dengan fitur yang dipilih membedakan suatu kriteria dari kriteria lain dalam node yang sama. Kefleksibelan ini meningkatkan kualitas keputusan dibandingkan dengan metode penghitungan satu tahap yang lebih konvensional.
- Dalam analisis multivariat, terutama dengan jumlah kriteria dan kelas yang tinggi, pohon keputusan dapat menghindari masalah dengan menggunakan kriteria yang lebih sedikit pada setiap node internal tanpa mengurangi kualitas keputusan.

2) Kekurangan Metode Pohon Keputusan

Adapun kekurangan dari penerapan algoritma *Decision Tree* adalah sebagai berikut:

- Terjadi tumpang tindih terutama ketika terdapat banyak kelas dan kriteria yang digunakan, yang dapat menyebabkan peningkatan waktu pengambilan keputusan dan penggunaan memori.
- Akumulasi kesalahan dari setiap tingkat dalam pohon keputusan yang besar.
- Kesulitan dalam merancang pohon keputusan yang optimal.
- Kualitas keputusan yang dihasilkan sangat tergantung pada desain pohon tersebut.

C. Klasifikasi

Dalam *Big Data Analytics*, pengumpulan data pada algoritma *decision tree* dapat dilakukan berdasarkan klasifikasi. Klasifikasi adalah pengelompokan dari suatu rekaman (set pelatihan), di mana setiap rekaman berisi kumpulan atribut. Tujuan klasifikasi adalah untuk menemukan model dari atribut kelas, sehingga kemudian dapat memberikan nilai terhadap atribut lainnya.

Saat mengklasifikasikan data, perhatian harus diberikan terhadap apakah data tersebut termasuk dalam data latihan atau data pengujian. Data latihan merupakan data yang memiliki proporsi lebih besar daripada data pengujian. Data-data tersebut kemudian diolah menggunakan bantuan komputer sehingga menghasilkan suatu model.

Terdapat beberapa metode klasifikasi, antara lain:

1. Metode berbasis Pohon Keputusan

   Contoh penerapan pohon keputusan adalah saat ingin mengklasifikasikan pengembalian suatu produk. Dengan teknik ini, kita dapat mengelompokkan kategori berdasarkan tingkat pengembalian, seperti ya/tidak, status dan penghasilan pajak. Hasilnya adalah suatu kesimpulan apakah uang tersebut layak dikembalikan atau tidak. Oleh karena itu, dalam pohon keputusan, sangat penting untuk membagi data menjadi data latihan yang berisi informasi yang sudah pasti, untuk kemudian memodelkan data lain dengan menggunakan data dari set data (data yang belum memiliki hasil akhir).

2. Dalam menghasilkan pohon keputusan, terdapat banyak algoritma yang dapat digunakan, seperti Hunt's, CART, ID3, C4.5, serta SLIQ SPRINT. Dalam proses induksi, beberapa hal yang perlu diperhatikan melibatkan:

   - Strategi yang Rakus

     Cara menentukan pembagian terbaik dengan menguji satu per satu

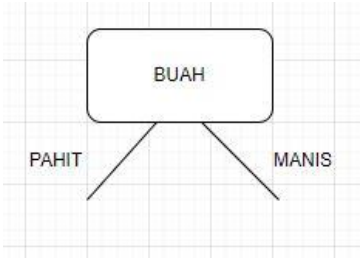
     data yang tersedia.

- Permasalahan dalam Menentukan Kondisi Uji

  Bagaimana menentukan kondisi uji, yang memerlukan perhatian apakah atribut data tersebut bersifat nominal, ordinal, atau kontinu.

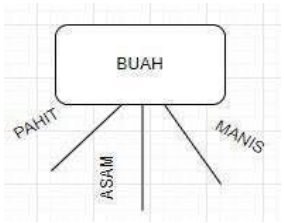
1) Nominal

Pada algoritma *Decision Tree* terdapat *Multi-way split dan binary split,* adalah dua pendekatan yang berbeda dalam proses pemisahan (*splitting*) data pada pohon keputusan (*decision tree*). Dalam *binary split*, pada setiap node keputusan, data dibagi menjadi dua cabang atau dua kelompok. Misalnya, pada tingkat suatu atribut, kita bisa membagi data menjadi dua kelompok berdasarkan kondisi tertentu, seperti ya/tidak, lebih besar/lebih kecil dari suatu nilai tertentu, dsb. Contohnya dapat dilihat pada gambar 4.



Gambar 4 Binary way split (Nominal)

Dalam *multi-way split*, pada setiap node keputusan, data dibagi menjadi lebih dari dua kelompok atau cabang. Ada tingkat suatu atribut, kita dapat membagi data menjadi tiga atau lebih kelompok berdasarkan kondisi tertentu. Contohnya pada Gambar 5.



Gambar 5 Multi way split (Nominal)

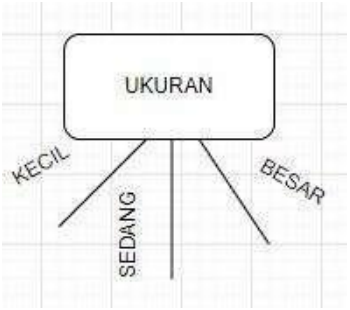
2) Ordinal

Dalam konteks *decision tree*, "ordinal" mengacu pada jenis atribut atau variabel yang memiliki tingkatan atau urutan tertentu, tetapi jarak antar tingkatnya tidak dapat diukur atau tidak memiliki interpretasi jarak

yang bermakna. Atribut ordinal memiliki nilai yang dapat diurutkan, tetapi perbedaan antar nilai tidak memiliki arti.

Contoh atribut ordinal dalam konteks *decision tree* bisa berupa kelas pendidikan, tingkat kepuasan (rendah, sedang, tinggi), atau peringkat posisi pekerjaan. Meskipun kita dapat mengurutkan nilai-nilai ini, perbedaan antara nilai-nilai tersebut tidak memiliki makna jarak yang bermakna. Contohnya *Binary way* dan *Multi way* dapat dilihat pada Gambar 6 dan 7 di bawah ini.

Gambar 6 Binary way split (Ordinal)



Gambar 7 Multi way split (Ordinal)

3) Continus

Dalam konteks *decision tree*, istilah *"continu"* atau *"continuous"* merujuk pada jenis atribut atau variabel yang memiliki nilai yang dapat diukur dalam skala kontinu. Atribut kontinu adalah atribut yang dapat mengambil sejumlah tak terbatas nilai dalam suatu rentang. Sebagai contoh, atribut kontinu dapat berupa tinggi badan, berat badan, suhu, atau pendapatan. Jadi, atribut kontinu adalah atribut yang memiliki nilai dalam skala kontinu, dan pemilihan split pada atribut ini adalah proses kritis dalam pembentukan decision tree untuk menentukan aturan keputusan yang optimal. Contoh *Continus Binary-way* dan *Multi-way* dapat dilihat pada Gambar 8 dan 9.

Gambar 8 Binary way split (Continus)

Gambar 9 Multi way split (Continus)

HASIL DAN PEMBAHASAN

A. *Dataset*

Pada penelitian ini yang menggunakan algoritma *Decision Tree*, kami memberikan gambaran cara kerjanya yaitu pada model yang ingin dibuat diharapkan dapat membuat klasifikasi pelanggan di suatu bank, apakah pelanggan tersebut berstatus aktif atau tidak aktif. Data yang kami gunakan pada penelitian ini adalah *dataset* customer. Namun pada *dataset* tersebut kami tidak melibatkan seluruh *feature*, hanya beberapa *feature* tertentu yang terlibat dalam penelitian ini, yaitu lokasi, id produk, nama pelanggan, saldo, dan status. *Training dataset* tersebut berisi dari **74.258 baris data**.

Berikut adalah *syntax* dan tampilan data yang kami olah menggunakan bahasa pemrograman Python.

```
In [1]: #impor library

        import pandas as pd
        df= pd.read_excel("data_customer1.xlsx")
        df.head()
```

Out[1]:

	loc	prdid	acnum	cusnum	name	acoff	open	kerja	lstrx	balance	sts
0	1	3	927414	126519	A '*****A 'I	3054	20160805	2	20160809	49517.21	2
1	7	15	960341	158554	A .*****UDIN	6065	20161108	2	20171227	1534052.81	1
2	3	15	958539	156915	AA*****AA	5641	20120326	5	20171206	631856.52	1
3	7	1	912961	112424	AA*****ADIS	4968	20121214	4	20121214	40000.00	2
4	8	17	966663	164151	AA*****AMAR	3500	20111207	1	20171221	23109.31	1

Gambar 10 Syntax Import library dan pemanggilan dataset

Dari data di atas, kita dapat melihat bahwa ada banyak *feature* yang tersedia. Namun, pada penelitian ini kami sepakat untuk tidak melibatkan seluruh *feature*, maka digunakan fungsi *drop()* untuk menghilangkan beberapa *feature* dan kembali disimpan pada *data frame* bernama **df**. Berikut ini merupakan *syntax* dan hasilnya:

```
In [2]: df = df.drop(['acnum','cusnum','acoff','lstrx','kerja','open'],axis=1)
        print(df)

             loc  prdid          name     balance  sts
        0      1      3  A '*****A 'I    49517.21    2
        1      7     15  A .*****UDIN  1534052.81    1
        2      3     15   A A*****A A   631856.52    1
        3      7      1  A A*****ADIS    40000.00    2
        4      8     17  A A*****AMAR    23109.31    1
        ...  ...    ...           ...         ...  ...
        74253  7      3  ZUL*****VIAN    48437.07    2
        74254  5      3  ZUR*****AIDA    49000.00    2
        74255  1      1  ZUV*****IBAR    40520.87    2
        74256  1     17  ZWA*****ANDA    19049.62    2
        74257  1      3  ZWE*****AN K  1824696.55    2

        [74258 rows x 5 columns]
```

Gambar 11 Syntax Data Frame

Pada tahap ini, *feature* sudah siap untuk lakukan pada tahap selanjutnya yaitu *training datasets*. Dalam penelitian ini, *feature* yang digunakan untuk training disimpan dalam variabel X (huruf besar) dan labelnya dalam variabel y (huruf kecil.

Berikut syntaxnya :

```
In [49]: # menyimpan feature untuk training dalam variabel X dan labelnya dalam variat
         X = df.drop(['name','sts'], axis=1) # feature
         y = df['sts']  #label
```

Gambar 12 Syntax Menyimpan feature dalam variable x dan y

Pada *syntax* diatas, terdapat *feature* 'nama' yang dibuang karena tidak diperlukan pada *training dataset* dan *feature* 'label' yang akan digunakan pada variabel y. Selanjutnya, sebelum menguji akurasi model, kita akan ambil 20% (dalam angka desimal menjadi 0.2) data dari *training dataset* sebagai *test dataset. Test Dataset* merupakan tahapan untuk menguji sebuah model yang dibuat setelah *training dataset* telah dilakukan. Berikut ini adalah *syntax* yang menggunakan fungsi *train_test*: **split().** Hasilnya akan disimpan pada variabel X_train dan y_train, sementara *test dataset* di X_test dan y_test.

```
In [50]: import sklearn.model_selection as ms
         X_train, X_test, y_train, y_test = ms.train_test_split(X,y, test_size=0.2)
```

Gambar 13 Syntax Test Dataset

B. Implementasi *Decision Tree*

Pada implementasi *decision tree* algoritma CART (*Classification and Regression Trees*) yang diterapkan pada Scikit-learn. Scikit-learn memiliki sebuah fungsi bernama *DecisionTreeClassifier* yang tergabung dalam paket *sklearn.tree* untuk mengimplementasikan *Decision Tree.* Pada uji coba yang pertama ini kami menggunakan *entropy* untuk melatih model sebagai kriteria pengukuran proses *splitting.*

Berikut ini syntax dan hasil yang diperoleh dari *entropy*:

```
In [6]: #pengukuran proses Splitting

        import sklearn.tree as tree
        model1 = tree.DecisionTreeClassifier(criterion='entropy', max_depth=5)
        model1.fit(X_train, y_train)

Out[6]: DecisionTreeClassifier
        DecisionTreeClassifier(criterion='entropy', max_depth=5)
```

Gambar 14 Syntax Pengukuran Proses Splitting

Pada tahap ini, model sudah terbangun dan selanjutnya akan dicoba untuk melakukan *test dataset.* Pada *test dataset*, data yang digunakan dari variabel X_test yang telah disiapkan sebelumnya dan hasil prediksinya disimpan pada variabel y_prediksi. Berikut ini gambar dari *syntax* dan hasilnya:

```
y_prediksi = model1.predict(X_test) #hasil prediksi disimpan di dataframe y_
```

Gambar 15 Syntax Variable y_prediksi

Hasilnya dapat kita lihat menggunakan fungsi **print(),** seperti pada gambar dibawah ini:

```
In [8]: print(y_prediksi)
        [2 2 2 ... 2 2 2]
```

Gambar 16 Syntax Print Function y_prediski

Karena datanya cukup banyak, jadi kita tidak bisa melihat seluruhnya. Namun, kita dapat melihat akurasi modelnya dengan menggunakan fungi akurasi yang disediakan di dalam sklearn seperti pada *syntax* dibawah ini:

```
import sklearn.metrics as met
print(met.accuracy_score(y_test, y_prediksi))
```

```
0.8911931053056827
```

Gambar 17 Syntax Print Akurasi Sklearn

Hasilnya yaitu 89%, itu berarti sudah cukup baik. Namun, untuk melihat hasil yang lebih akurat, kita juga bisa melakukan perbandingan dengan mengganti kriteria *splitting* dari *entropy* menjadi *gini index* dan Kembali lakukan proses akurasinya dengan cara yang sama.
Berikut hasil akurasi menggunakan *gini index*:

```
In [10]: # mengganti kriteria dari entropy menjadi gini index

import sklearn.tree as tree
import sklearn.metrics as met
model1 = tree.DecisionTreeClassifier(criterion='gini', max_depth=5)
model1.fit(X_train, y_train)
y_prediksi = model1.predict(X_test)
print(y_prediksi)
print(met.accuracy_score(y_test, y_prediksi))
```

```
[2 2 2 ... 2 2 2]
0.8913950983032588
```

Gambar 18 Syntax mengganti Kriteria dari entropy menjadi gini index

Ternyata hasilnya adalah 89% yang artinya kurang lebih sama dengan *entropy*, maka kita dapat menggunakan *entropy* ataupun *gini index* pada test dataset ini.

C. Visualisasi Hasil Penelitian

Pada penggunaan model *Decision Tree* untuk melihat informasi lebih spesifik, maka kita perlu membuat visualialisasi. Visual dalam model ini berbentuk diagram yang menyerupai pohon (*tree*) dimana tingkatan paling atas bernama *root* hingga bagian yang tidak memiliki cabang yang bernama *leaf* . Dalam menampilkan visualisasi dari *decision tree*, kita menggunakan library **Graphviz** dan **Pydotplus.** Untuk mengunduh kedua modul tersebut, kita melakukan perintah *install* pada conda, seperti berikut ini :

```
conda install -c anaconda graphviz

Retrieving notices: ...working... done
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done
```

Gambar 19 Syntax Instalasi Paket Graphiz

```
conda install -c conda-forge pydotplus

Note: you may need to restart the kernel to use updated packages.Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working...
Warning: 2 possible package resolutions (only showing differing packages):
  - anaconda/win-64::openssl-1.1.1w-h2bbff1b_0
  - defaults/win-64::openssl-1.1.1w-h2bbff1b_0done
```

Gambar 20 Syntax Instalasi Paket pydotplus

Setelah keduanya berhasil di-*install*, untuk menampilkan *Decision Tree* kita dapat memanggil dengan fungsi **export_graphviz()** dengan memasukkan data dari 'model1' sebagai salahsatu parameternya, dan menggunakan **labels** sebagai parameter lainnya dari semua *feature* yang ada (yaitu loc, prdid, dan balance).

Output dari fungsi ini adalah **dot_data** yang berisi data berformat khusus DOT yang dipakai oleh Graphviz. Hasil visualisasi kita menggunakan fungsi **write_png()** menjadi sebuah gambar (berformat PNG) dan tersimpan dalam folder lokal komputer. Hasilnya output kita beri nama **decisiontree_produk.png**.

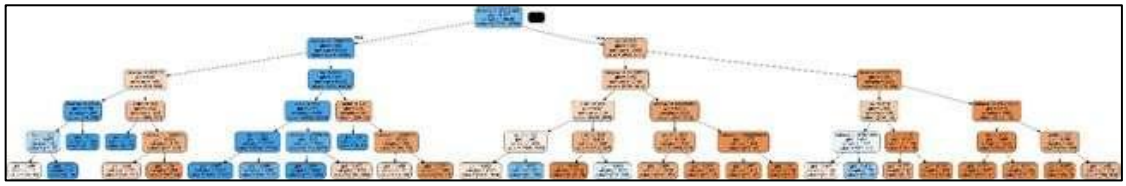
Berikut merupakan perintah dan hasil dari visualisasi *decision tree*:

```
import pydotplus as pp
labels = ['loc','prdid','balance']
dot_data = tree.export_graphviz(model1, out_file=None, feature_names=labels, filled=True, rounded=True)
graph = pp.graph_from_dot_data(dot_data)
graph.write_png('decisiontree_produk.png') #hasilnya disimpan dalam bentuk png
```

Gambar 21 Syntax Visualisasi Decision Tree

Bila kita membuka file PNG, maka akan muncul visualisasinya sebagai

berikut :



Gambar 22 Visualisasi Decision Tree

Dari diagram di atas dapat memberikan informasi yang lebih baik dan lengkap untuk kita. Informasi yang disajikan mencakup *feature* yang diuji di setiap node, nilai *gini*, jumlah 'sampel', dan jumlah elemen data di setiap kelas data. Pada bagian *root node*, *feature* yang dipilih oleh algoritma untuk diuji adalah jumlah saldo (*balance*) apakah dibawah dari 50 juta, lalu hasil *splitting* pertama menghasilkan kelompok True yang berisi 46.988 sampel, kelompok False membuat uji lagi dari *feature* lokasi (*loc*) ada berapa data di lokasi yang dibawah 32, dan menghasilkan *splitting* baru yang dapat dilihat pada gambar visualisasi. Hingga akhirnya menghasilkan sebuah *leaf node* dari masing-masing *splitting* untuk bisa dijadikan keputusan.

KESIMPULAN

Pengenalan algoritma pohon keputusan untuk mengklasifikasikan nasabah aktif atau tidak aktif dalam data perbankan secara signifikan meningkatkan pemahaman tentang perilaku nasabah dan optimalisasi strategi pemasaran. Keberhasilan algoritma ini terletak pada kemampuannya menghasilkan aturan keputusan yang mudah diinterpretasikan, memberikan gambaran jelas tentang pendorong keterlibatan pelanggan dan mendukung pengambilan keputusan yang lebih cerdas. Algoritma pohon keputusan telah terbukti efektif dalam memprediksi apakah klien aktif atau tidak aktif berdasarkan data yang dikumpulkan. Analisis pohon keputusan dapat mengidentifikasi faktor-faktor penting seperti saldo rekening, frekuensi transaksi dan jenis produk atau layanan yang digunakan. Pemahaman yang mendalam terhadap variabel-variabel tersebut memungkinkan bank merancang strategi pemasaran yang lebih tepat sasaran. Salah satu keunggulan utama aplikasi ini adalah kemampuannya dalam mengoptimalkan strategi pemasaran. Dengan mengetahui faktor-faktor yang mempengaruhi keterlibatan nasabah, bank dapat menyesuaikan pendekatannya untuk meningkatkan retensi nasabah. Misalnya, mereka mungkin menargetkan pelanggan dengan saldo rekening tertentu atau menawarkan produk khusus kepada kelompok pelanggan tertentu. Hal ini tidak hanya meningkatkan kepuasan nasabah, namun juga dapat meningkatkan pendapatan dan profitabilitas bank.Selain itu, model pohon keputusan memberikan

pemahaman yang lebih baik tentang logika keputusan. Hal ini membantu pemangku kepentingan, termasuk tim manajemen dan pemasaran, membuat keputusan yang lebih tepat.

Dengan mengetahui faktor-faktor kritis tersebut, bank dapat mengambil langkah-langkah strategis untuk meningkatkan operasional bisnis dan memberikan pengalaman nasabah yang lebih baik. Namun perlu diingat bahwa evaluasi dan pemeliharaan model pohon keputusan harus dilakukan secara berkala. Dengan memperbarui data nasabah dan mengadaptasi model terhadap perubahan tren atau perilaku nasabah, bank dapat memastikan bahwa model tersebut relevan dan akurat. Hal ini juga membantu menghadapi dinamika pasar yang terus berubah. Penerapan algoritma pohon keputusan untuk mengklasifikasikan nasabah aktif dan tidak aktif dalam data perbankan membawa manfaat nyata dalam mengoptimalkan strategi bisnis. Dengan memahami perilaku nasabah dengan lebih baik, bank dapat meningkatkan retensi nasabah, memperluas basis nasabah, dan mencapai kesuksesan jangka panjang dalam industri yang sangat kompetitif ini.

REFERENSI

Firdaus Uus, Utama N Ditdit. 2020. *"Balance as one of the attributes in the customer segmentation analysis method : Systematic literature review"*. *Advances in Science, Technology and Engineering System Journal* (ASTES), 5(2). http://www.astesj.com/

Feby, Dita. *"Apa Itu Decision Tree di Machine Learning Model?"* https://dqlab.id. Diakses pada 14 Januari 2024. https://dqlab.id/apa-itu-decision-tree-di-machine-learning-model

Puspitorini I, Sintawati D Ita. 2021. "Penerapan *Data Mining* untuk Klasifikasi Prediksi Produk Jenis Makanan Kucing yang Sesuai Kebutuhan dengan Algoritma *Decision Tree* (ID3)". Jurnal AKRAB Juara, 6(4), hal 21-26. http://dx.doi.org/10.58487/akrabjuara.v6i4.1629

Wadhwatanya. (2021). "*Splitting Criteria and Algorithms in Decision Tree*" wadhwaatanya1234.medium.com. Diakses pada 17 Januari 2024. https://wadhwatanya1234.medium.com/decision-tree-model-1089c11ab22

Sudibyo, A., Asra, T., & Rifai, B. (2018). "Menggunakan Metode Algoritma *Decision Tree"*. 14(2), 145. http://nusamandiri.ac.id/aji.aby@nusamandiri.ac.id http://bsi.ac.id http://nusamandiri.ac.id/

Kurniawan D. (2020). "Pengenalan *Machine Learning* dengan Python". Jakarta : PT Elex Media Komputindo.