Tugas Besar 2 IF3170 Inteligensi Artifisial
Implementasi Algoritma Pembelajaran Mesin

Disusun Oleh:

- Daniel Mulia Putra Manurung (13522043)
- Benjamin Sihombing (13522054)
- Adril Putra Merin (13522068)
- Berto Richardo Togatorop (13522118)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

DAFTAR ISI

DAFTAR ISI **2**

BAB I
PENJELASAN SINGKAT MODEL **2**

- Penjelasan Singkat Implementasi KNN 2
- Penjelasan singkat Implementasi Naive-Bayes 2
- Penjelasan Singkat Implementasi ID3 2

BAB II
TAHAP CLEANING DAN PREPROCESSING **2**

BAB III
HASIL PREDIKSI **3**

PEMBAGIAN TUGAS **4**

REFERENSI **4**

BAB I PENJELASAN SINGKAT MODEL

- **Penjelasan Singkat Implementasi KNN**

  KNN (K - Nearest Neighbors) adalah model atau algoritma *machine learning* yang memanfaatkan *neighbors* dalam prediksinya. KNN tidak melakukan proses apa-apa pada saat *training*, KNN hanya menyimpan data saja. Pada saat *predict*, KNN akan mencari *neighbors* dari input dengan memilih k titik (*instance* data) dari data *train* yang memiliki jarak terpendek dengan input. Untuk menghitung jarak, ada beberapa metric yang digunakan. Pada umumnya metric yang digunakan adalah Minkowski *distance*, khususnya Euclidean *distance* (Minkowski *distance* dengan p=2) dan Manhattan *distance* (Minkowski *distance* dengan p=2).

$$d(x, y) = \sqrt[p]{\sum_{i=1}^{n} (x_i - y_i)^p},$$

$$n\ dimension$$

Eq 1. Minkowski *distance*

Setelah menemukan k *neighbors*, akan dilakukan *voting* terhadap k *neighbors* tersebut. *Voting* yang dilakukan adalah *voting* label. Dari hasil *voting,* label dengan jumlah terbanyak akan menjadi prediksi label dari input.

- **Penjelasan singkat Implementasi Naive-Bayes.**

  Naive-Bayes atau Gaussian Naive-Bayes adalah algoritma machine learning berbasis probabilitas yang memprediksi label berdasarkan Teori Bayes. Rumus Teori Bayes untuk klasifikasi dirumuskan sebagai:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

Eq 2. Teori Bayes untuk klasifikasi

Pada tahap training, model menghitung:

1. Prior probability($P(v_j)$), yaitu peluang terjadinya label v_j.
2. Likelihood ($P(a_i|v_j)$), yaitu peluang terjadinya atribut ai berdasarkan label v_j, yang dihitung menggunakan distribusi normal Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Eq 3. Likelihood untuk Gaussian Naive Bayes

Pada tahap prediksi, model menghitung probabilitas posterior $P(v_j|a_1, a_2,..., a_n)$ untuk setiap kelas dan memprediksi label berdasarkan nilai probabilitas posterior tertinggi.

Pada implementasi Gaussian Naive-Bayes, klasifikasi ditentukan berdasarkan probabilitas posterior. Langkah-langkah yang dilakukan dalam pembentukan model Gaussian Naive-Bayes adalah sebagai berikut:

1. Menghitung Statistik Data Latih
- Untuk setiap kelas dalam data target (label), hitung mean dan variansi untuk setiap fitur dalam data latih.
- Hitung prior probability setiap kelas berdasarkan proporsi jumlah data di kelas tersebut terhadap total data latih.

2. Menghitung Likelihood
   Untuk setiap fitur pada data uji, likelihood dihitung menggunakan formula distribusi normal Gaussian. Parameter distribusi normal, yaitu mean (μ) dan variansi(σ^2), diambil dari hasil langkah sebelumnya.

3. Menggunakan Log Probability
   Untuk meningkatkan stabilitas numerik dan menghindari underflow saat mengalikan probabilitas kecil, perhitungan dilakukan di log-space. Underflow terjadi ketika angka yang sangat kecil (lebih kecil dari yang bisa direpresentasikan oleh komputer) diubah menjadi nol dalam representasi floating-point. Logaritma mengubah perkalian menjadi penjumlahan. Konstanta (yang berasal dari P(x)) tidak mempengaruhi pemilihan kelas, sehingga sering diabaikan. Akhirnya, probabilitas posterior logaritmik dirumuskan sebagai:

$$\log P(y|x) = \log P(y) + \sum_i \log P(x_i|y)$$

Eq 4. probabilitas posterior dalam log-space

4. Klasifikasi Data Uji
- Hitung probabilitas posterior untuk setiap kelas berdasarkan log prior dan log likelihood.
- Prediksi kelas dengan probabilitas posterior tertinggi sebagai hasil klasifikasi.

5. Melakukan Prediksi
   Ulangi langkah perhitungan posterior untuk setiap sampel dalam data uji.

- **Penjelasan Singkat Implementasi ID3**

  ID3 adalah salah satu algoritma untuk mengimplementasikan model *Decision Tree Learning*. Untuk membentuk *decision tree*, ID3 melakukan langkah berikut:

  1. Menghitung *Information Gain* dari setiap fitur.

$$IG(S, A) = Entropy(S) - \sum((|S_v| / |S|) * Entropy(S_v))$$

Eq 4. Information Gain Formula

$$Entropy(S) = - \sum p_i * \log_2(p_i) \; ; \; i = 1 \text{ to } n$$

Eq 5. Entropy Formula

  2. Jika label dari setiap data masih berbeda, pisahkan data dan bentuk *node* baru berdasarkan nilai fitur yang memiliki nilai *information gain* tertinggi.
  3. Jika node memiliki label yang sama untuk semua data, *node* menjadi *leaf node* dengan nilai label tersebut.
  4. Ulangi proses ini, sampai semua fitur telah habis atau sampai seluruh node tidak bisa menghasilkan *child* lagi.

Jika terdapat sebuah instance dalam data yang mana pada tree ID3 tidak terdapat klasifikasi maka class dari instance tersebut akan ditentukan menggunakan probabilistic prediction. Yang mana akan dilakukan pemilihan berdasarkan frekuensi kemunculan target kelas.

BAB II
TAHAP CLEANING DAN PREPROCESSING

Data Cleaning

Handling Missing Data

Kelas TransformMissingValue adalah *transformer* yang dirancang khusus untuk menangani nilai hilang dalam dataset menggunakan *library* scikit-learn. Karena kelas ini adalah *inheritance dari* BaseEstimator dan TransformerMixin, kelas ini bisa dimasukkan ke dalam pipeline *machine learning* untuk tahap preprocessing data. TransformMissingValue mengatasi *missing values*, baik numerik maupun kategorikal, dengan strategi pengisian yang berbeda-beda untuk tiap jenis kolom. Contohnya pada data ini numerik dapat diisi menggunakan strategi median, sementara data kategorikal diisi dengan nilai default yang ditentukan.

```
class TransformMissingValue(BaseEstimator, TransformerMixin):
  def __init__(self, proto_strategy = 'any'):
    # self.knn_imputer = KNNImputer()
    self.proto_strategy = proto_strategy
    self.knn_imputer = SimpleImputer(missing_values=np.nan, strategy='median')
    self.median_imputer = SimpleImputer(missing_values=np.nan, strategy='median')
    self.median_cols = ['sttl', 'dttl', 'swin', 'dwin', 'trans_depth', 'ct_state_ttl', 'ct_flw_http_mthd
    self.knn_cols = ['dur', 'sbytes', 'dbytes', 'sloss', 'dloss', 'sload', 'dload', 'spkts', 'dpkts', 's
    self.categorical_cols = ['proto', 'state', 'service', 'is_sm_ips_ports', 'is_ftp_login']

  def fit(self, X, y=None):
    self.knn_imputer.fit(X[self.knn_cols])
    self.median_imputer.fit(X[self.median_cols])
    return self

  def transform(self, X):
    X = X.copy()
    X[self.knn_cols] = self.knn_imputer.transform(X[self.knn_cols])
    X[self.median_cols] = self.median_imputer.transform(X[self.median_cols])
    if 'proto' in X.columns:
      fill_value = 'any' if self.proto_strategy == 'any' else '-'
      X['proto'].fillna(fill_value, inplace=True)

    if 'state' in X.columns:
      X['state'].fillna('no', inplace=True)

    if 'service' in X.columns:
      X['service'].fillna('-', inplace=True)

    if 'is_sm_ips_ports' in X.columns:
      X['is_sm_ips_ports'].fillna(0, inplace=True)

    if 'is_ftp_login' in X.columns:
      X['is_ftp_login'].fillna(0, inplace=True)
    return X
```

Median_imputer digunakan karena datanya memiliki *skewness* yang tinggi (bahkan lebih mirip L). Data yang bernilai besar akan lebih mendominasi. Untuk kolom kategorikal, kami mengisi nilai default berdasarkan observasi yang kami lakukan untuk tiap kolom tersebut. Contohnya pada kolom ‘proto’, kami menggunakan nilai default ‘any’ yang menandakan bahwa protokolnya tidak dibatasi. Pada beberapa kolom, kami juga

menambahkan kategori baru, yaitu ‘no’ atau ‘-’ yang menandakan bahwa tidak ada service atau state yang dimiliki.

Dealing With Outliers

Untuk menangani *Outliers*, kami tidak membuang data *outlier*. Tapi, kami lebih memilih untuk melakukan proses *clipping*. Proses *clipping* adalah proses mengubah data *outlier* menjadi nilai maksimum dan minimum. Hal ini dilakukan karena kami menganggap ada data penting yang terbuang jika kami membuang fitur maupun row. Kolom-kolom yang diterapkan *clipping* adalah 'sbytes', 'dbytes', 'sloss', 'dloss', 'sload', 'dload', 'spkts', 'dpkts', 'response_body_len', 'sjit', 'djit', 'sinpkt', 'dinpkt', 'tcprtt', 'synack', 'ackdat', 'dur', 'stcpb', 'dtcpb', 'smean', 'dmean'. Selain clipping, kami juga melakukan log transformation untuk mengurangi skewness pada data ini yang memiliki distribusi right-skewed. Hal ini berguna untuk algorithma ML yang mengasumsikan hubungan linier atau distribusi normal serperti KNN, regresi, dll.

Remove Duplicates

Untuk menangani data yang duplikat, kami membuang seluruh data yang duplikat. Tidak ada alasan khusus dalam keputusan ini, namun data duplikat memang harus dihapus karena dapat mengganggu kinerja model.

Feature Engineering

Pada proses feature engineering, kami memilih fitur terbaik untuk melatih model dengan menggunakan teknik SelectKBest, yang memilih k fitur terbaik berdasarkan mutual information, guna mengurangi noise dalam data. Kelas FeatureEngineering menerapkan metode ini untuk memilih fitur yang paling relevan dengan target, meningkatkan efisiensi model.

Selain itu, kelas ID3FeatureEngineering menggunakan KBinsDiscretizer untuk mendiskretisasi kolom numerik menjadi kategori diskrit. Pendekatan ini cocok untuk algoritma berbasis pohon keputusan, yang lebih efektif dengan data kategorikal, sementara kolom kategorikal tetap tidak berubah.

Kombinasi kedua teknik ini mengoptimalkan pemilihan fitur dan mempersiapkan data untuk model berbasis pohon keputusan, meningkatkan akurasi dan interpretabilitas model dengan mengurangi pengaruh noise.

Data Preprocessing

Feature Scaling

```
class FeatureScaling(BaseEstimator, TransformerMixin):
  def __init__(self):
    self.standard_scaler = StandardScaler()
    self.min_max_scaler = MinMaxScaler()
    self.standard_cols = ['dur', 'sbytes', 'dbytes', 'sloss', 'dloss', 'sload', 'dload', 'spkts',
'dpkts', 'stcpb', 'dtcpb', 'smean', 'dmean', 'response_body_len', 'sjit', 'djit', 'sinpkt',
'dinpkt', 'tcprtt', 'synack', 'ackdat']
    self.min_max_cols = ['sttl', 'dttl', 'swin', 'dwin', 'trans_depth', 'ct_state_ttl',
'ct_flw_http_mthd', 'ct_ftp_cmd', 'ct_srv_src', 'ct_srv_dst', 'ct_dst_ltm', 'ct_src_ltm',
'ct_src_dport_ltm', 'ct_dst_sport_ltm', 'ct_dst_src_ltm', 'proto_encoded',
'proto_target_encoded', 'state_target_encoded', 'service_target_encoded']
    self.standard_cols_used = []
    self.min_max_cols_used = []
```

```
def fit(self, X, y=None):
  self.standard_cols_used = [col for col in self.standard_cols if col in X.columns]
  self.min_max_cols_used = [col for col in self.min_max_cols if col in X.columns]

  self.standard_scaler.fit(X[self.standard_cols_used])
  self.min_max_scaler.fit(X[self.min_max_cols_used])
  return self

def transform(self, X):
  X = X.copy()

  if self.standard_cols_used:
      X[self.standard_cols_used] = self.standard_scaler.transform(X[self.standard_cols_used])

  if self.min_max_cols_used:
      X[self.min_max_cols_used] = self.min_max_scaler.transform(X[self.min_max_cols_used])

  return X
```

Pada tahap feature scaling, kolom-kolom dalam dataset dibagi berdasarkan metode yang paling sesuai dengan karakteristik datanya. Kolom-kolom dengan nilai kontinu yang cenderung memiliki distribusi right-skewed, seperti dur, sbytes, sload, dan dload, diproses menggunakan StandardScaler. Metode ini menormalkan data ke skala z-score dengan mean 0 dan standar deviasi 1, sehingga lebih cocok untuk fitur-fitur dengan rentang nilai yang luas. Sementara itu, kolom-kolom dengan rentang nilai yang terbatas atau memiliki ordinalitas tinggi, seperti sttl, dttl, proto_encoded, dan state_target_encoded, diproses menggunakan MinMaxScaler, yang menormalkan nilai ke dalam rentang [0, 1]. Pendekatan ini memastikan semua fitur memiliki skala yang seragam, sehingga membantu meningkatkan performa algoritma machine learning tanpa mengubah informasi penting dalam data.

Feature Encoding

- KNN

  Pada model KNN, feature encoding dilakukan dengan pendekatan yang disesuaikan dengan karakteristik data. Untuk fitur proto, digunakan frequency encoding karena memiliki tingkat ordinalitas yang cukup tinggi. Pendekatan ini mengganti nilai kategori dengan proporsi kemunculannya di data, sehingga menjaga informasi distribusi frekuensi. Sementara itu, fitur state dan service diubah menggunakan one-hot encoding karena merupakan data nominal dengan ordinalitas rendah. Dengan cara ini, setiap kategori pada state dan service direpresentasikan sebagai kolom biner, sehingga lebih sesuai dengan kebutuhan model KNN yang sensitif terhadap skala dan jarak antar fitur.

- Gaussian Naive Bayes

  Pada preprocessing untuk model Gaussian Naive Bayes (GNB), dilakukan target encoding pada fitur proto, state, dan service. Pendekatan ini dipilih karena target encoding mengubah kategori menjadi nilai kontinu berdasarkan hubungan dengan target label. Hal ini sangat sesuai untuk GNB yang mengasumsikan input fitur berupa data kontinu dan mengikuti distribusi Gaussian, sehingga encoding ini membantu model memanfaatkan informasi dari data secara lebih efektif.

- ID3

Pada model dengan FeatureEncodingID3, encoding fitur kategorikal dilakukan menggunakan Ordinal Encoding. Fitur seperti proto, state, service, is_sm_ips_ports, dan is_ftp_login yang bersifat kategorikal, dikonversi menjadi nilai numerik dengan mempertahankan urutan atau relasi antar kategori. Teknik ini dipilih karena beberapa fitur memiliki hubungan ordinal, seperti pada proto yang menggambarkan urutan protokol. Prosesnya dimulai dengan melatih encoder menggunakan fit() untuk mempelajari kategori pada fitur yang ditentukan, lalu mengubah data dengan transform() sehingga fitur kategorikal menjadi numerik. Pendekatan ini sederhana namun efektif untuk menangani data kategorikal dalam model.

Dimensionality Reduction

Pada kelas PCATransform, preprocessing dilakukan dengan menggunakan Principal Component Analysis (PCA) untuk reduksi dimensi data. PCA digunakan untuk mengurangi jumlah fitur dalam dataset dengan menjaga sebagian besar variasi data, sehingga meningkatkan efisiensi model. Pada kelas ini, parameter n_components ditentukan untuk memilih jumlah komponen utama yang ingin dipertahankan, dengan default 0.95 yang berarti 95% dari variansi data akan dipertahankan. Prosesnya dimulai dengan fit(), yang melatih PCA pada data untuk menemukan komponen utama, dan diikuti oleh transform() yang mengubah data menjadi ruang dimensi yang lebih rendah berdasarkan komponen utama yang dipelajari. Teknik ini sangat bermanfaat untuk meningkatkan kinerja model dengan mengurangi kompleksitas dan potensi overfitting.

BAB III
HASIL PREDIKSI

F-1 Score

	From Scratch	Library
KNN	0.5177014578322122	0.5188767078146481
GNB	0.3072088410933842	0.3072088410933842
DTL ID3	0.4160476706858228	0.4341364509669988

Accuracy

	From Scratch	Library
KNN	0.7873050272320283	0.7838261712623684
GNB	0.6039236932903704	0.6039236932903704
DTL ID3	0.7476973965610654	0.7553394736091705

Precision

	From Scratch	Library
KNN	0.568385356381558	0.5732511437150916
GNB	0.391447926187862	0.391447926187862
DTL ID3	0.4457208554693433	0.4614790836048973

Recall

	From Scratch	Library
KNN	0.4975024625805385	0.4968226944805812
GNB	0.33853200784238513	0.33853200784238513
DTL ID3	0.4118527290983717	0.4275180154405683

Berdasarkan hasil eksperimen kami, didapatkan bahwa hasil implementasi algoritma kami untuk KNN, GNB, DTL ID3 memiliki hasil yang sangat menyerupai hasil dari library yang digunakan. Hal ini mengimplikasikan bahwa implementasi algoritma yang kami hasilkan cukup akurat untuk menjadi pemodel hasil.

KNN

Algoritma KNN mempunyai hasil yang lebih baik dibandingkan GNB dan ID3 karena beberapa hal. Pertama, KNN lebih baik pada data yang tidak non-linear. Dari hasil EDA, data tidak terdistribusi normal dan tidak

linear. Hal ini membuat KNN lebih baik dibandingkan GNB yang beranggap datanya terdistribusi normal (gauss). KNN juga lebih baik dibandingkan 2 algoritma lainnya ketika jumlah data dan dimensinya cenderung besar. Terakhir, dengan algoritma KNN, kami bisa mengubah nilai k dan p untuk bereksperimen mencari model yang terbaik.

Gaussian Naive Bayes

Algoritma GNB menghasilkan performa yang paling buruk dibandingkan KNN dan ID3. Ada satu alasan besar yang menyebabkan GNB memiliki performa yang buruk pada data ini. Algoritma GNB menggunakan distribusi gauss yang berarti menganggap data terdistribusi normal. Namun, pada fakta, data sangat jauh dari terdistribusi normal. Datanya terdistribusi secara beragam setiap fiturnya. Ada yang memiliki *skewness* yang tinggi sekali (mirip L). Ada yang memiliki 2 *skewness* kiri dan kanan, namun tengahnya kosong. Ada yang terdistribusi *uniform*. Dari fitur, tidak ada satupun yang terdistribusi normal. Oleh karena itu, GNB memiliki hasil yang sangat buruk pada data ini

ID3

Algoritma ID3 memberikan hasil yang tidak sebaik algoritma KNN, yaitu pada kisaran angka 0.4. Algoritma ID3 memiliki beberapa kekurangan yang membuatnya tidak efektif dalam berbagai situasi. Yang pertama adalah algoritma ID3 sangat mungkin untuk melakukan overfitting. ID3 cenderung membuat pohon keputusan yang terlalu kompleks, sesuai dengan training data, sehingga memiliki hasil generalisasi yang buruk pada data yang belum pernah dilihat sebelumnya. Ini terutama terjadi ketika data mengandung banyak noise atau pohon yang dibangun tanpa adanya pemangkasan atau pruning. Kedua adalah tidak adanya pruning. Algoritma ID3 tidak menggunakan teknik pemangkasan, yang bertujuan untuk menghasilkan cabang - cabangyang tidak diperlukan dalam pembuatan pohon keputusan. Tanpa perlakuan pruning, ID3 menghasilkan pohon yang sangat besar dan spesifik sehingga sulit untuk diinterpretasikan dan overfit. Yang berikutnya adalah algoritma ID3 memiliki bias terhadap atribut yang memiliki nilai lebih banyak. ID3 menggunakan information gain untuk memilih atribut terbaik untuk membagi data, yang biasanya bias terhadap atribut dengan nilai kategori yang lebih banyak, walaupun atribut tersebut tidak memberikan pembagian yang paling impactful. Dengan keterbatasan - keterbatasan ini, meskipun algoritma ID3 dapat memberikan pohon keputusan yang cepat dan sederhana, algoritma ini kurang efektif dalam menghadapi data yang kompleks atau data yang memiliki banyak atribut dan nilai kontinu, seperti dataset yang kita gunakan dalam tugas ini.

PEMBAGIAN TUGAS

NIM	Task
13522043	ID3 Modelling
13522054	KNN Modelling
13522068	Data Cleaning and Preprocessing
13522118	Gaussian Naive Bayes Modelling

REFERENSI

K-Nearest Neighbor(KNN) Algorithm - GeeksforGeeks
Bayes Theorem | Statement, Formula, Derivation, and Examples
PPT Materi Naive Bayes Mata Kuliah AI Teknik Informatika ITB
Naive Bayes Sklearn
Addressing Numerical Underflow in Naive Bayes Classification
ID3 Implementation