Tugas Besar II IF3170 Inteligensi Artifisial
Implementasi Algoritma Pembelajaran Mesin

Kelompok Learning Tin Can

Justin Aditya Putra Prabakti (13522130)
Attara Majesta Ayub (13522139)
Jason Fernando (13522156)
Atqiya Haydar Luqman (13522163)

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

Daftar Isi

Implementasi K-Nearest Neighbors

Implementasi KNN cukup singkat. Pertama, dilakukan iterasi untuk setiap row pada data test yang diberikan, dihitung *distance*-nya dengan semua item dalam data training, lalu disimpan N tetangga dengan nilai paling rendah (paling dekat dengan row acuan).

Sebelum melakukan implementasi, KNN akan mengecek apakah data yang diberi sama dengan data sebelumnya untuk melewati perhitungan. Hal ini dilakukan karena KNN sedikit lambat dan harus mengulang kalkulasi setiap kali dipanggil. Model KNN memiliki 2 fungsi : **guess** dan **indices**. Guess hanya mengembalikkan hasil terbaik (modus) dari N tetangga, sedangkan indices mengembalikan persentase nya.

```
testknn = KNN(10,"manhattan")
testknn.fit(X_red, y_red)


result = testknn.guess(Xt_red)


indicenn = testknn.indices(Xt_red)
```

```
result[:10]
```

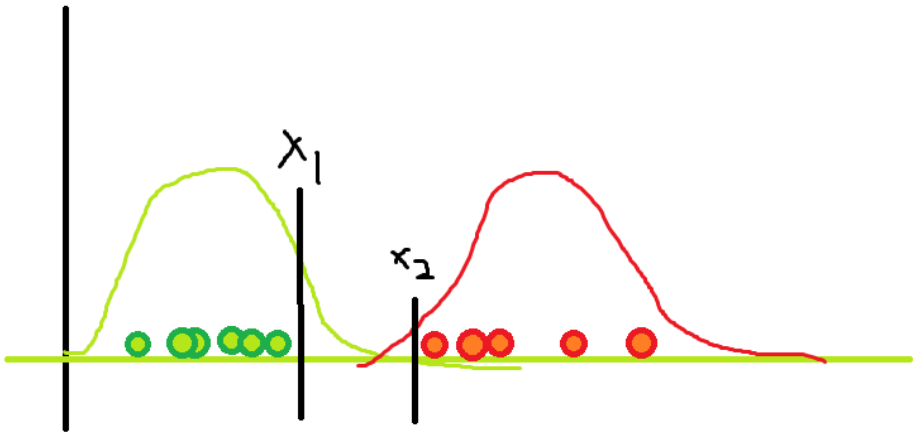
```
['Exploits',
 'Exploits',
 'Exploits',
 'Exploits',
 'Analysis',
 'DoS',
 'DoS',
 'DoS',
 'Exploits',
 'Normal']
```

```
indicenn.head()
```

	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance	Shellcode	Worms
0	0.0	0.0	0.3	0.5	0.0	0.1	0.0	0.1	0.0	0.0
1	0.0	0.0	0.1	0.5	0.1	0.0	0.3	0.0	0.0	0.0
2	0.1	0.0	0.1	0.6	0.0	0.0	0.2	0.0	0.0	0.0
3	0.2	0.0	0.0	0.6	0.0	0.0	0.2	0.0	0.0	0.0
4	0.5	0.0	0.1	0.2	0.0	0.0	0.2	0.0	0.0	0.0

Implementasi Naive-Bayes

Naive-bayes diimplementasikan menggunakan pendekatan "Gaussian Naive Bayes", dimana "probabilitas" dari suatu kolom tergantung kepada jarak atau posisinya pada distribusi normal (*also known as Gaussian distribution, hence gaussian*) sesuai untuk kelas tersebut.



Implementasi ID3

Implementasi algoritma ID3 dilakukan dengan membangun decision tree secara rekursif. Pertama, entropi dihitung untuk mengukur impurity dari data target, lalu information gain digunakan untuk menentukan atribut terbaik yang akan digunakan untuk split data. Proses ini terus berlanjut hingga semua label target dalam subset sama, atribut yang tersedia habis, kedalaman maksimal (max_depth) tercapai, atau jumlah sampel dalam node kurang dari batas minimum (min_samples_split).

Untuk setiap pemisahan, subset data dibagi berdasarkan nilai atribut yang terpilih, dan proses rekursi dilanjutkan hingga memenuhi salah satu kondisi berhenti. Setelah tree selesai dibangun, pruning dapat dilakukan (jika diaktifkan) dengan cara memangkas subtree yang tidak meningkatkan akurasi model. Tree yang dihasilkan dapat digunakan untuk melakukan prediksi dengan cara menavigasi tree berdasarkan nilai-nilai atribut pada data baru. Implementasi ini fleksibel, dengan parameter seperti max_depth dan min_samples_split untuk mengontrol kompleksitas dan kinerja model.

Cleaning dan Preprocessing

1. Value Reductor

Untuk membantu proses *under-fitting*, dibuat fungsi ValueReductor(), yang mengurangi jumlah sampel, namun tetap menjaga distribusi label pada sampel. ValueReductor() memiliki parameter: reduction_percent, prioritize_na, na_hit_rate. Berikut adalah penjelasannya

- Reduction_percent
  Sisa berapa persen dari data asli yang diinginkan
- Prioritize_na
  Dahulukan null values untuk pengurangan sebanyak na_hit_rate
- Na_hit_rate
  Jika prioritize_na true, maka menunjukkan rasio dari jumlah "pengurangan" null value yang secara eksplisit dikurangi (Contoh : na_hit_rate 0.50 dan redcution_rate 0.50 untuk tabel dengan 100 rows, berarti dari 50 (100 * 0.50) rows yang akan dihapus, diusahakan semuanya null dulu, baru dari nilai biasa jika tidak cukup)

2. Feature Imputer

Untuk menangani missing values dalam dataset, FeatureImputer mengisi nilai kosong pada kolom numerik dan kategorikal menggunakan metode most frequent.

Parameter

- num_imputer
  Menggunakan SimpleImputer dengan strategi most_frequent untuk kolom numerik.
- cat_imputer
  Menggunakan SimpleImputer dengan strategi most_frequent untuk kolom kategorikal.

Missing values pada kolom numerik diisi dengan nilai yang paling sering muncul. Missing values pada kolom kategorikal diisi dengan nilai yang paling sering muncul.

```
from sklearn.impute import SimpleImputer

class FeatureImputer(BaseEstimator, TransformerMixin):
    def __init__(self):
        pass

    def fit(self, X, y=None):
        return self

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

        num_columns = X.select_dtypes(include='number').columns
        cat_columns = X.select_dtypes(include='object').columns

        # Impute missing values in numerical columns
        num_imputer = SimpleImputer(strategy='most_frequent')
        X[num_columns] = num_imputer.fit_transform(X[num_columns])

        # Impute missing values in categorical columns
        cat_imputer = SimpleImputer(strategy='most_frequent')
        X[cat_columns] = cat_imputer.fit_transform(X[cat_columns])

        return X
```

3. Outlier Remover

OutlierRemover digunakan untuk mendeteksi dan menangani nilai ekstrem pada kolom numerik dengan membatasi nilai hingga persentil ke-95.

Parameter

- exclude_columns

  Daftar kolom yang tidak akan diproses untuk outlier.

Jika nilai maksimum di kolom numerik lebih dari 10 kali median dan lebih besar dari 10, nilai tersebut dibatasi hingga nilai persentil ke-95.

```
import pandas as pd
import numpy as np
from sklearn.base import BaseEstimator, TransformerMixin

class OutlierRemover(BaseEstimator, TransformerMixin):
    def __init__(self):
        pass

    def fit(self, X, y=None):
        return self
```

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

        # Select numerical columns, excluding specified columns
        numerical_columns = X.select_dtypes(include=['float64',
'int64']).columns.tolist()
        exclude_columns = ['sttl', 'dttl', 'dwin', 'stcpb', 'dtcpb']
        numerical_columns = [col for col in numerical_columns if col not in
exclude_columns]

        for feature in numerical_columns:
            max_value = X[feature].max()
            median_value = X[feature].median()
            if max_value > 10 * median_value and max_value > 10:
                upper_bound = X[feature].quantile(0.95)
                X[feature] = np.where(X[feature] < upper_bound, X[feature],
upper_bound)

        return X
```

4. Duplicate Remover

DuplicateRemover menghapus baris duplikat pada dataset. Baris yang memiliki semua nilai yang sama akan dihapus menggunakan fungsi drop_duplicates() dari pandas.

```
class DuplicateRemover(BaseEstimator, TransformerMixin):
    def __init__(self):
        pass

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        X = X.copy()
        X.drop_duplicates(inplace=True)
        return X
```

5. Feature Engineer

FeatureEngineering menyederhanakan kategori dengan frekuensi rendah pada kolom kategorikal. Jika jumlah kategori unik lebih dari 4, hanya 3 kategori teratas yang dipertahankan, dan kategori lainnya diubah menjadi simbol '-'.

```
import numpy as np
import pandas as pd
from sklearn.base import BaseEstimator, TransformerMixin

class FeatureEngineering(BaseEstimator, TransformerMixin):
    def __init__(self):
        pass

    def fit(self, X, y=None):
        return self

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

        # Select categorical columns
        categorical_columns =
X.select_dtypes(include=['object']).columns.tolist()

        for feature in categorical_columns:
            if X[feature].nunique() > 4:
                top_categories = X[feature].value_counts().head(3).index
                X[feature] = np.where(X[feature].isin(top_categories),
X[feature], '-')

        return X
```

6. Feature Dropper

FeatureDropper digunakan untuk menghapus kolom tertentu yang tidak relevan.

Parameter

- drop_cols
  Daftar kolom yang akan dihapus.

```
class FeatureDropper(BaseEstimator, TransformerMixin):
    def __init__(self, drop_cols: List[str]):
        self.drop_cols = drop_cols

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        X = X.copy()
        X.drop(columns=self.drop_cols, inplace=True)
        return X
```

7. Feature Transformer

FeatureTransformer melakukan transformasi logaritmik pada kolom numerik dengan banyak nilai unik (>50). FeatureTransformer melakukan transformasi logaritmik pada kolom numerik dengan banyak nilai unik (>50).

```
import numpy as np
import pandas as pd
from sklearn.base import BaseEstimator, TransformerMixin

class FeatureTransformer(BaseEstimator, TransformerMixin):
    def __init__(self):
        pass

    def fit(self, X, y=None):
        return self

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

        # Select numerical columns, excluding specified columns
        numerical_columns =
X.select_dtypes(include=[np.number]).columns.tolist()

        for feature in numerical_columns:
            # Check the number of unique values
            if X[feature].nunique() > 50:
                # Apply logarithmic transformation
                if X[feature].min() == 0:
                    X[feature] = np.log(X[feature] + 1)
                else:
                    X[feature] = np.log(X[feature])

        return X
```

8. Feature Encoder

FeatureEncoder melakukan one-hot encoding pada kolom kategorikal tertentu.

Parameter

- columns
  Daftar kolom yang akan di-encode.

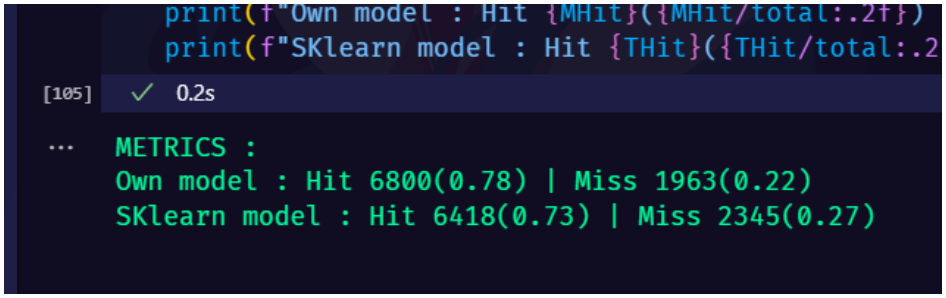
Kolom kategorikal diubah menjadi beberapa kolom biner menggunakan metode one-hot encoding.

```
import pandas as pd
```

```
import numpy as np
from sklearn.base import BaseEstimator, TransformerMixin

class FeatureEncoder(BaseEstimator, TransformerMixin):
    def __init__(self, columns):
        self.columns = columns

    def fit(self, X, y=None):
        return self

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

        # Lakukan one-hot encoding hanya pada kolom tertentu
        X_encoded = pd.get_dummies(X, columns=self.columns, drop_first=False)

        return X_encoded
```

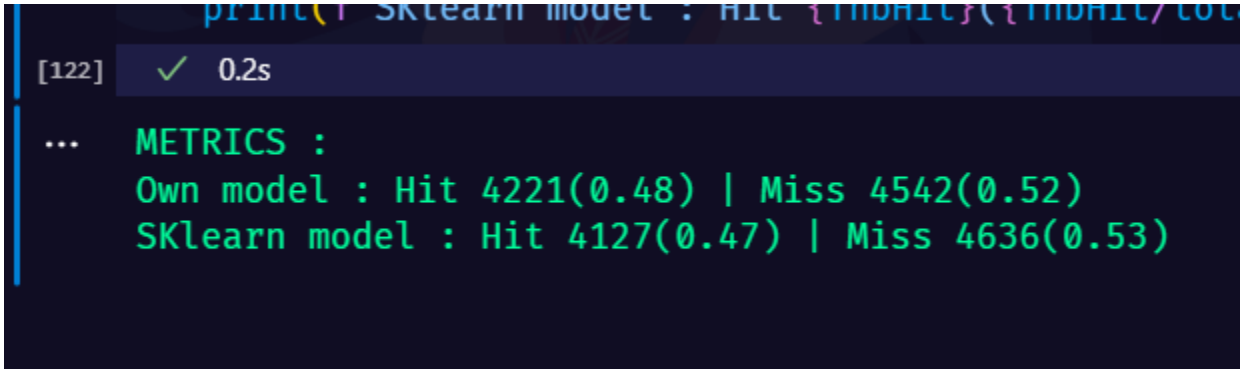
Perbandingan Hasil Prediksi

KNN (Manhattan, 10 Nearest Neighbor)



Menggunakan ValueReductor(25) untuk mempercepat perhitungan dan sedikit Underfitting

NaiveBayes



Menggunakan ValueReductor(25) untuk mempercepat perhitungan dan sedikit Underfitting

```
Hit: 3563 (0.40659591464110467) // Miss: 5200 (0.5934040853588953)
```

Kontribusi

13522130	ValueReductor, KNN, NaiveBayes
13522139	ID3, Improvements, Fix Error, Error Analysis
13522156	ID3, Improvements, Fix Error
13522163	EDA, Data Cleaning, Preprocessing, Pipeline, Split Dataset

Referensi

K-Nearest-Neighbors:

- Nishom, M. "Perbandingan Akurasi Euclidean Distance, Minkowski Distance, dan Manhattan Distance pada Algoritma K-Means Clustering Berbasis Chi-Square." *Jurnal Informatika Politeknik Harapan Bersama*, vol. 4, no. 1, 30 Jan. 2019, pp. 20-24, doi:10.30591/jpit.v4i1.1253. Diakses melalui https://www.neliti.com/publications/466869/perbandingan-akurasi-euclidean-distance-minkowski-distance-dan-manhattan-distanc pada 15 Desember 2024

Naive bayes :

- https://youtu.be/yRzIyWVEaCQ?si=lnFMMRQf3AcKEQ8N Diakses pada 15 Desember 2024
- https://medium.com/@kashishdafe0410/gaussian-naive-bayes-understanding-the-basics-and-applications-52098087b963 Diakses pada 15 Desember 2024