LAPORAN MATA KULIAH MACHINE LEARNING

KNN (K-NEAREST NEIGHBOR)

Dosen Pengampu :

Prof. Dr. Enny Itje Sela, S. Si., M. Kom.

Disusun oleh :

Pingkan Putri N – 5220411173

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS & TEKNOLOGI

UNIVERSITAS TEKNOLOGI YOGYAKARTA

2024

I. ALGORITMA KNN

Algoritma K-Nearest Neighbor (K-NN) adalah metode klasifikasi yang memanfaatkan informasi dari tetangga terdekat untuk mengategorikan objek baru berdasarkan atribut dan sampel pada data latihan. Prosesnya dimulai dengan mengukur jarak antara objek yang akan diprediksi dengan semua objek dalam set data latihan, dan kemudian memilih K tetangga terdekat. Dalam konteks ini, misalnya, kita dapat membayangkan situasi di mana pemerintah ingin menentukan apakah suatu rumah berada di wilayah Kota Yogyakarta atau Kabupaten Sleman.

Langkah pertama adalah menentukan jumlah tetangga yang akan diperhitungkan (k), sebagai contoh, mungkin memilih 3 tetangga terdekat (k = 3). Langkah kedua melibatkan perhitungan jarak antara rumah yang akan diprediksi dan rumah-rumah di sekitarnya, diurutkan dari yang terkecil hingga yang terbesar. Selanjutnya, langkah ketiga melibatkan pemilihan 3 (k) tetangga terdekat tersebut, di mana label mayoritas dari tetangga tersebut akan menjadi prediksi. Misalnya, jika 2 dari 3 tetangga termasuk dalam wilayah Kota Yogyakarta, maka prediksi akan menyatakan bahwa rumah tersebut juga termasuk dalam wilayah Kota Yogyakarta; sebaliknya, jika 2 dari 3 tetangga termasuk dalam wilayah Kabupaten Sleman, maka prediksi akan menyatakan bahwa rumah tersebut termasuk dalam wilayah Kabupaten Sleman.

II. CONTOH KASUS KNN

Contoh kasus ini menggunakan dataset Iris dengan menyertakan Python dengan library scikit-learn. Berikut adalah langkah-langkah yang digunakan dalam metode K-Nearest Neighbor:

1. Impor Library:

   Pertama kita harus mengimpor beberapa library yang diperlukan dari scikit-learn, seperti datasets untuk memuat dataset, train_test_split untuk membagi data menjadi data latih dan uji, KNeighborsClassifier untuk membuat model KNN, dan metrics untuk mengukur performa model.

```
[7]  # Impor library
     from sklearn import datasets
     from sklearn.model_selection import train_test_split
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn import metrics
```

2. Memuat Dataset Iris:

   Di sini kita menggunakan dataset Iris sebagai contoh. Dataset ini berisi informasi tentang tiga spesies iris (setosa, versicolor, dan virginica).

```
# Memuat dataset iris
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

3. Membagi Data:

   train_test_split digunakan untuk membagi dataset menjadi data latih dan data uji. Dalam contoh ini, 80% data digunakan untuk melatih model, dan 20% digunakan untuk menguji model.

```
[9] # Membagi data menjadi data latih dan data uji
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

4. Membuat Model KNN:

   Model KNN dibuat menggunakan KNeighborsClassifier dengan parameter n_neighbors=3, yang berarti model akan mempertimbangkan tiga tetangga terdekat.

```
[10] # Membuat model KNN
     knn = KNeighborsClassifier(n_neighbors=3)
```

5. Melatih Model:

   Model KNN dilatih menggunakan data latih (X_train dan y_train).

```
[11] # Melatih model
     knn.fit(X_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

6. Melakukan Prediksi:

   Model yang telah dilatih digunakan untuk melakukan prediksi pada data uji (X_test), menghasilkan y_pred, yang merupakan prediksi kelas untuk setiap sampel.

```
[12] # Melakukan prediksi pada data uji
     y_pred = knn.predict(X_test)
```

7. Evaluasi Performa Model:

   Performa model dievaluasi menggunakan metrik akurasi (metrics.accuracy_score) dengan membandingkan prediksi model (y_pred) dengan label sebenarnya (y_test). Di sini kita juga dapat melihat akurasi dari perhitungan data yang telah diuji.

```
[13] # Evaluasi performa model
     accuracy = metrics.accuracy_score(y_test, y_pred)
     print(f'Akurasi: {accuracy}')

     Akurasi: 1.0
```

8. Contoh Prediksi Baru:

   Di sini saya juga menambahkan sebuah contoh buah baru dengan atribut [5.1, 3.5, 1.4, 0.2] yang akan diberikan sebagai input untuk model yang telah dilatih. Model akan memprediksi kategori buah baru tersebut, dan hasil prediksi dicetak ke layar. Dan hasil yang muncul dari hasil test menggunakan data baru adalah kategori setosa.

```
# Contoh prediksi buah baru
new_data = [[5.1, 3.5, 1.4, 0.2]]  # Contoh atribut buah baru
prediction = knn.predict(new_data)
print(f'Prediksi kategori: {iris.target_names[prediction[0]]}')

Prediksi kategori: setosa
```

III. PROGRAM KNN PADA BAB 2

Nilai K	Test_size	Jarak	Akurasi (%)
7	0,3	Euclidian	0.97777
11	0,3	Euclidian	0.97777
71	0,3	Euclidian	0,68889
17	0,3	Euclidian	0,97777
3	0,2	Euclidian	1,0
7	0,2	Euclidian	0,96667
11	0,2	Euclidian	0,96667
71	0,2	Euclidian	0,83334
7	0,3	Manhattan	0,95556
17	0,3	Manhattan	0,97777
3	0,3	Manhattan	0,97777
11	0,3	Manhattan	0,95556
71	0,3	Manhattan	0,95556
7	0,2	Manhattan	0,96667
11	0,2	Manhattan	0,96667
71	0,2	Manhattan	0,83334

Berdasarkan tabel ini, akan dihitung akurasi menggunakan Program KNN seperti yang ada pada bab 2. Dengan menambah beberapa nilai K dan jumlah data test. Berikut

penjelasannya:

1. Memanggil Dataset Iris:

   Kode ini memuat dataset Iris dari sklearn.datasets dan menyimpannya dalam variabel `iris_ku`.

```
[101] # memanggil dataset iris
      from sklearn.datasets import load_iris
      iris_ku = load_iris()
```

2. Menyimpan Data Fitur dan Target:

   Variabel `X` menyimpan fitur (atribut) dari dataset, dan variabel `y` menyimpan target (label).

```
[102] # simpan data fitur/kolom (X) dan target (y)
      X = iris_ku.data
      y = iris_ku.target
```

3. Menyimpan Nama Fitur dan Target:

   Variabel `feature_names` dan `target_names` menyimpan nama fitur dan target dataset.

```
[103] # Simpan nama fitur/kolom (X) dan target (y)
      feature_names = iris_ku.feature_names
      target_names = iris_ku.target_names
```

4. Tampilkan Nama Fitur dan Target:

   Kode ini mencetak nama fitur dan target dataset.

```
# tampil nama fitur dan target dataset
print("Feature names:", feature_names)
print("Target names:", target_names)

Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Target names: ['setosa' 'versicolor' 'virginica']
```

5. Tampilkan Tipe Data dan 5 Baris Pertama dari Fitur (X):

   Kode ini mencetak tipe data dari `X` (biasanya numpy array) dan lima baris pertama dari fitur.

```
[105] # X dan y adalah numpy arrays
      print("\nType of X is:", type(X))

      Type of X is: <class 'numpy.ndarray'>
```

```
[106] # tampilkan 5 baris pertama
      print("\nFirst 5 rows of X: \n", X[:5])

First 5 rows of X:
 [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
```

6. Pembagian Data Latih dan Uji:

   Kode ini membagi dataset menjadi data latih dan uji.

```
[108] # splitting X dan y untuk data latih dan uji
      from sklearn.model_selection import train_test_split
      X_latih, X_tes, y_latih, y_tes = train_test_split(X, y, test_size=0.3, random_state=1)
```

7. Pelatihan Model KNN:

   Kode ini membuat model KNN dengan jumlah tetangga terdekat (`n_neighbors`) sebanyak 3, dan melatih model menggunakan data latih.

```
# pelatihan pada data latih menggunakan KNN (k=3)
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=7, metric='minkowski')
knn.fit(X_latih, y_latih)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=7)
```

8. Prediksi pada Data Uji:

   Kode ini melakukan prediksi menggunakan model KNN pada data uji.

```
[112] # melakukan prediksi pada data uji
      y_prediksi = knn.predict(X_tes)
```

9. Evaluasi Akurasi Model:

   Kode ini mencetak akurasi model KNN dengan membandingkan nilai aktual (`y_tes`) dengan nilai prediksi (`y_prediksi`).

```
[113] # perbandingan nilai aktual (y_tes) dengan nilai prediksi (y _prediksi)
      from sklearn import metrics
      print ("Akurasi model kNN:", metrics.accuracy_score(y_tes, y_prediksi))

      Akurasi model kNN: 0.9777777777777777
```

10. Prediksi Menggunakan Data Contoh Baru:

    Kode ini melakukan prediksi menggunakan data contoh baru dan mencetak hasil prediksi.

```
[114] # prediksi menggunakan data sampel dibuat sendiri
      contoh = [[3, 5, 4, 2], [2, 3, 5, 4]]
      preds = knn.predict (contoh)
      pred_species = [iris_ku.target_names[p] for p in preds]
      print ("Prediksi :", pred_species)

      Prediksi : ['versicolor', 'virginica']
```

IV. PROGRAM KNN PADA BAB 3

Pada bab 3 saya akan menggunakan data cuaca untuk menguji datanya menggunakan program KNN. Berikut adalah data yang digunakan:

cuaca1	suhu	kelembaban	angin	main
overcast	hot	high	FALSE	yes
overcast	cool	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
rainy	mild	normal	FALSE	yes
rainy	mild	high	TRUE	no
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
sunny	mild	normal	TRUE	yes

Menggunakan program KNN dengan nilai K = 2, data di atas akan diuji. Berikut adalah langkah-langkah yang saya gunakan untuk menguji data di atas:

Kode ini dapat dijelaskan langkah demi langkah sebagai berikut:

1. Membaca Data dari File Excel:

   Menggunakan pandas untuk membaca data dari file Excel ke dalam DataFrame dengan nama `dataku`.

```
[18] import pandas as pd
     dataku = pd.read_excel('/content/drive/MyDrive/Colab Notebooks/Dataset/cuaca.xlsx')
```

2. Menampilkan Informasi Data:

   Code ini menampilkan ukuran DataFrame (jumlah baris dan kolom), nama kolom (fitur) yang ada dalam DataFrame, dan DataFrame secara keseluruhan.

```
[19] print("Ukuran:", dataku.shape)
     Ukuran: (14, 5)
[20] print("InFeatures:", dataku.columns)
     InFeatures: Index(['cuaca1', 'suhu', 'kelembaban', 'angin', 'main'], dtype='object')
```

dataku

	cuaca1	suhu	kelembaban	angin	main
0	overcast	hot	high	False	yes
1	overcast	cool	normal	True	yes
2	overcast	mild	high	True	yes
3	overcast	hot	normal	False	yes
4	rainy	mild	high	False	yes
5	rainy	cool	normal	False	yes
6	rainy	cool	normal	True	no
7	rainy	mild	normal	False	yes
8	rainy	mild	high	True	no
9	sunny	hot	high	False	no
10	sunny	hot	high	True	no
11	sunny	mild	high	False	no
12	sunny	cool	normal	False	yes
13	sunny	mild	normal	True	yes

3. Label Encoding:

   Menggunakan LabelEncoder untuk mengubah nilai-nilai dalam setiap kolom menjadi nilai numerik.

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for col in dataku.columns:
  dataku[col] = le.fit_transform(dataku[col])

dataku
```

	cuaca1	suhu	kelembaban	angin	main
0	0	1	0	0	1
1	0	0	1	1	1
2	0	2	0	1	1
3	0	1	1	0	1
4	1	2	0	0	1
5	1	0	1	0	1
6	1	0	1	1	0
7	1	2	1	0	1
8	1	2	0	1	0
9	2	1	0	0	0
10	2	1	0	1	0
11	2	2	0	0	0
12	2	0	1	0	1
13	2	2	1	1	1

4. Pembagian Data menjadi Fitur (X) dan Target (y):

   Memisahkan fitur (X) dan target (y) dari DataFrame.

```
[24] X = dataku[dataku.columns[:-1]]
     y = dataku[dataku.columns[-1]]
```

5. Menampilkan Fitur dan Target:

   Menampilkan matriks fitur (X) dan vektor target (y).

```
print("\nFeature matrix: \n", X)
print("\Response vector: \n", y)
```

```
Feature matrix:
     cuaca1  suhu  kelembaban  angin
0         0     1           0      0
1         0     0           1      1
2         0     2           0      1
3         0     1           1      0
4         1     2           0      0
5         1     0           1      0
6         1     0           1      1
7         1     2           1      0
8         1     2           0      1
9         2     1           0      0
10        2     1           0      1
11        2     2           0      0
12        2     0           1      0
13        2     2           1      1
```

```
\Response vector:
 0     1
1      1
2      1
3      1
4      1
5      1
6      0
7      1
8      0
9      0
10     0
11     0
12     1
13     1
Name: main, dtype: int64
```

6. Pembagian Data Latih dan Data Uji:

   Membagi data menjadi data latih dan data uji menggunakan train_test_split.

```
[39] from sklearn.model_selection import train_test_split
     X_latih, X_tes, y_latih, y_tes = train_test_split(X, y, test_size=0.3, random_state=1)
```

7. Pembuatan Model k-NN dan Pelatihan:

   Menggunakan algoritma k-NN (k-Nearest Neighbors) dengan k=3 dan menggunakan jarak Manhattan sebagai metrik. Pelatihan model dilakukan pada data latih.

```
# pelatihan pada data latih menggunakan KNN (k=3)
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_latih, y_latih)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

8. Prediksi pada Data Uji:

   Menggunakan model yang telah dilatih untuk melakukan prediksi pada data uji.

```
# melakukan prediksi pada data uji
y_prediksi = knn.predict(X_tes)
```

9. Evaluasi Model:

   Menggunakan metrik akurasi untuk mengevaluasi sejauh mana model k-NN dapat memprediksi dengan benar pada data uji. Hasil akurasi dicetak.

```
# perbandingan nilai aktual (y_tes) dengan nilai prediksi (y _prediksi)
from sklearn import metrics
print ("Akurasi model kNN:", metrics.accuracy_score(y_tes, y_prediksi))
```

```
Akurasi model kNN: 0.8
```

   Disediakan juga data uji yang dapat diprediksi menggunakan program KNN di atas. Datanya sebagai berikut:

Nilai K	Test_size	Akurasi (%)
5	0,3	0,6
7	0,5	0,66666
7	0,3	0,6
3	0,7	0,2