LAPORAN OBSERVASI ANALISIS, DESAIN, DAN IMPLEMENTASI ALGORITMA KNN

Ditujukan untuk memenuhi salah satu tugas mata kuliah:
Sistem Cerdas (**CTI2G3-IT-43-03**)

oleh:

Bagas Alfito Prismawan (1303193027)
Kevin Antonio Fajrin (1303193123)
Sultan Kautsar (1303194010)

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2021

DAFTAR ISI

ABSTRAK

Dalam statistik, algoritma k-nearest neighbor adalah metode klasifikasi non-parametrik yang pertama kali dikembangkan oleh Evelyn Fix dan Joseph Hodges pada tahun 1951, dan kemudian diperluas oleh Thomas Cover. Ini digunakan untuk klasifikasi dan regresi. Dalam kedua kasus, input terdiri dari k contoh pelatihan terdekat dalam kumpulan data. Algoritma K-Nearest Neighbor (KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek yang berdasarkan dari data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Mengklasifikasikan objek baru berdasarkan atribut dari data train. Algoritma KNN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari sampel uji yang baru.

Kata kunci: KNN, Klasifikasi, Ketetanggaan

ABSTRACT

In statistics, the k-nearest neighbor algorithm is a non-parametric classification method that was first developed by Evelyn Fix and Joseph Hodges in 1951, and later expanded by Thomas Cover. It is used for classification and regression. In both cases, the input consists of the k closest training examples in the data set. The K-Nearest Neighbor (KNN) algorithm is a method for classifying objects based on the learning data that is closest to the object. Classifies new objects based on attributes from the train data. The KNN algorithm uses neighboring classification as the predictive value of the new test sample.

Keywords: KNN, Classification, Neighborhood

BAB I
PENDAHULUAN

A. Algoritma KNN

Prinsip kerja K-Nearest Neighbor (KNN) adalah mencari jarak terdekat antara data yang akan dievaluasi dengan k tetangga (neighbor) terdekatnya dalam data pelatihan(training). Dengan k merupakan banyaknya tetangga terdekat.

Data training diproyeksikan ke ruang berdimensi banyak, yang mana masing-masing dimensi menjelaskan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi data training. Sebuah titik pada ruang ini ditandai kelas k(titik hitam), kelas k merupakan klasifikasi yang paling banyak ditemui pada k buah tetangga terdekat titik tersebut.

Dalam menentukan nilai atau kelas k, sebaiknya kita gunakan nilai ganjil, karena jika tidak, ada kemungkinan kita tidak akan mendapatkan jawaban. Penentuan nilai k dipertimbangkan berdasarkan banyaknya data yang ada dan ukuran dimensi yang dibentuk oleh data. Semakin banyak data yang ada, angka k yang dipilih sebaiknya semakin rendah. Namun, semakin besar ukuran dimensi data, angka k yang dipilih sebaiknya semakin tinggi.

Untuk mencari dekat atau jauhnya jarak antar titik pada kelas k biasanya dihitung menggunakan jarak Euclidean. Jarak Euclidean adalah formula untuk mencari jarak antara 2 titik dalam ruang dua dimensi.

B. Langkah Kerja Algoritma KNN

- Tentukan parameter K = jumlah dari persekitaran (nearest neighbors)
- Hitung jarak antara data baru yang ditanyakan dengan seluruh sampel data pelatihan
- Urutkan seluruh jarak berdasarkan jarak minimum dan tetapkan persekitaran sesuai dengan nilai K.
- Sesuaikan klasifikasi dari kategori Y dengan persekitaran yang telah ditetapkan
- Gunakan kelas dengan jumlah terbanyak sebagai dasar menentukan kelas dari data baru yang ditanyakan.

C. Permasalahan

Diberikan DataSetTB3.xls yang berisikan sheet Data dan Submit untuk kasus hand-writing digit recognizer (bagian dari MNIST data yang diunduh pada www.kaggle.com). Data itu merepresentasikan gambar grayscale dari tulisan tangan dengan dimensi 28 x 28 pixel (total 784 pixel). Setiap pixel dapat berisi nilai 0-255 yang merepresentasikan derajat keabuan dari pixel, semakin tinggi semakin gelap.

Sheet Data terdiri atas 1001 baris dan 786 kolom, dimana baris pertama merupakan label attribute yaitu idData, label-kelas dan idPixel. Sheet Submit terdiri atas 786 kolom dengan kolom pertama adalah idData, kolom kedua adalah label kelas dari data 0-9 dan kolom lainnya merepresentasikan nilai pixel dengan rentang 0-255 (dalam 1 vektor).

Sheet Submit terdiri atas 501 baris dan 785 kolom, baris pertama merupakan label attribute yaitu idData dan idPixel. Sheet Submit tidak memiliki label kelas hanya mengandung nilai pixel yang disimpan dalam 785 kolom.

BAB II
DASAR TEORI

A. Handwritten Digit Recognition

Handwritten Digit Recognition atau pengenalan angka tulisan tangan adalah kemampuan komputer untuk mengenali angka tulisan tangan manusia. Ini adalah tugas yang sulit untuk mesin karena angka tulisan tangan tidak sempurna dan dapat dibuat dengan berbagai rasa.

MNIST problem adalah kumpulan data yang dikembangkan oleh Yann LeCun, Corinna Cortes dan Christopher Burges untuk mengevaluasi model pembelajaran mesin pada masalah klasifikasi digit tulisan tangan.

Dataset dibangun dari sejumlah dataset dokumen yang dipindai yang tersedia dari National Institute of Standards and Technology (NIST). Dari sinilah nama dataset berasal, sebagai Modified NIST atau MNIST dataset.

Gambar digit diambil dari berbagai dokumen yang dipindai, dinormalisasi dalam ukuran dan dipusatkan. Ini menjadikannya kumpulan data yang sangat baik untuk mengevaluasi model, memungkinkan pengembang untuk fokus pada pembelajaran mesin dengan sedikit pembersihan atau persiapan data yang diperlukan.

Setiap gambar berukuran 28 x 28 piksel persegi (total 784 piksel). Pemisahan standar kumpulan data digunakan untuk mengevaluasi dan membandingkan model, di mana 60.000 gambar digunakan untuk melatih model dan kumpulan 10.000 gambar terpisah digunakan untuk mengujinya.

Ini adalah tugas pengenalan digit. Dengan demikian ada 10 digit (0 hingga 9) atau 10 kelas untuk diprediksi. Hasil dilaporkan menggunakan kesalahan prediksi, yang tidak lebih dari akurasi klasifikasi terbalik.

Hasil yang sangat baik mencapai kesalahan prediksi kurang dari 1%. Kesalahan prediksi canggih sekitar 0,2% dapat dicapai dengan Jaringan Saraf Konvolusi yang besar. Ada daftar hasil mutakhir dan tautan ke makalah yang relevan di MNIST dan kumpulan data lainnya di halaman web Rodrigo Benenson.

B. Algoritma KNN

K-Nearest Neighbor adalah suatu metode yang menggunakan algoritma supervised learning dimana hasil dari instance yang baru diklasifikasikan berdasarkan mayoritas dari kategori k-tetangga terdekat. Algoritma k-Nearest Neighbor menggunakan Neighborhood Classification sebagai nilai prediksi dari nilai instance yang baru.

Prinsip kerja K-Nearest Neighbor (KNN) adalah mencari jarak terdekat antara data yang akan dievaluasi dengan k tetangga (neighbor) terdekatnya dalam data pelatihan(training) . Dengan k merupakan banyaknya tetangga terdekat.

Data training diproyeksikan ke ruang berdimensi banyak, yang mana masing-masing dimensi menjelaskan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi data training. Sebuah titik pada ruang ini ditandai kelas k(titik hitam) , kelas k merupakan klasifikasi yang paling banyak ditemui pada k buah tetangga terdekat titik tersebut.

BAB III
IMPLEMENTASI

A. Algoritma KNN

Terdapat Sheet Data yang terdiri atas 1001 baris dan 786 kolom, dimana baris pertama merupakan label attribute yaitu idData, label-kelas dan idPixel. Sheet Data terdiri atas 786 kolom dengan kolom pertama adalah idData, kolom kedua adalah label kelas dari data 0-9 dan kolom lainnya merepresentasikan nilai pixel dengan rentang 0-255 (dalam 1 vektor).

Sheet Submit terdiri atas 501 baris dan 785 kolom, baris pertama merupakan label attribute yaitu idData dan idPixel. Sheet Submit tidak memiliki label kelas hanya mengandung nilai pixel yang disimpan dalam 785 kolom.

- Sheet Submit digunakan untuk pengujian akhir ("pengetahuan terbaik" dari kNN yang telah diujikan secara lokal)
- Menggunakan sheet Data untuk melakukan pelatihan dan uji performansi lokal

B. Load dan Read Data Sheet

Kami memilih menggunakan library Pandas untuk membaca data yang berformat .xlsx dengan dibagi menjadi 2 sheet yaitu sheet submit dan sheet data

```
data_train = pd.read_excel('DataSetTB3_SHARE.xlsx', sheet_name='Data')
data_test = pd.read_excel('DataSetTB3_SHARE.xlsx', sheet_name='Submit')
```

C. Pre-Processing dan Split Data

Pada process ini algoritma akan mengecek jika nilai yang terdapat memiliki nilai null maka akan dilakukan proses normalisasi

```
miss_values = data_train.columns[data_train.isnull().any()]
print(f"Missing values:\n{data_train[miss_values].isnull().sum()}")

null_values = data_train.columns[data_train.isna().any()]
print(f"Null Values:\n{data_train[null_values].isna().sum()}")
```

Dikarenakan terdapat data latih yang null maka dilakukan proses normalisasi dengan algoritma sebagai berikut

```
columns = [ 'label', ['pixel'+str(i) for i in range (785)]]
scale_X = StandardScaler()
X = pd.DataFrame(scale_X.fit_transform(data_train.drop(["label"], axis=1),), columns = columns[1])

X.head(20)

y = pd.DataFrame(data_train, columns = ['label'])

from sklearn.model_selection import  train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state = 42, stratify = y)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

D. Pembangunan Model KNN

Algoritma berikut adalah proses berjalannya algoritma library knn pada setiap metrik.

```
# Modeling dan Train
# euclidean
knn = KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, metric='euclidean')
knn.fit(X_train, y_train)

# Manhattan
knn1: KNeighborsClassifier = KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30,
                                                  metric='manhattan')
knn1.fit(X_train, y_train)

# Chebyshev
knn2 = KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, metric='chebyshev')
knn2.fit(X_train, y_train)

# Minkowski
knn3 = KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, metric='minkowski')
knn3.fit(X_train, y_train)
```

E. Evaluation

Proses berikut adalah proses hasil keluaran dari algoritma KNN yang berupa Euclidean, Manhattan, Chebyshev, dan Minkowski.

```
# Evaluation
# Euclidean
y_pred_knn = knn.predict(X_test)

knn = metrics.confusion_matrix(y_test, y_pred_knn)
print(knn)
print(classification_report(y_test, y_pred_knn))

# Manhattan
y_pred_knn1 = knn1.predict(X_test)

knn11 = metrics.confusion_matrix(y_test, y_pred_knn1)
print(knn11)
print(classification_report(y_test, y_pred_knn1))

# Chebyshev
y_pred_knn2 = knn2.predict(X_test)

knn2 = metrics.confusion_matrix(y_test, y_pred_knn2)
print(knn2)
print(classification_report(y_test, y_pred_knn2))
```

```
# Minkowski
y_pred_knn3 = knn3.predict(X_test)

knn3 = metrics.confusion_matrix(y_test, y_pred_knn3)
print(knn3)
print(classification_report(y_test, y_pred_knn3))

accuracy1 = metrics.accuracy_score(y_test, y_pred_knn)
accuracy2 = metrics.accuracy_score(y_test, y_pred_knn1)
accuracy3 = metrics.accuracy_score(y_test, y_pred_knn2)
accuracy4 = metrics.accuracy_score(y_test, y_pred_knn3)

print("Euclidean:", accuracy1)
print("Manhattan:", accuracy2)
print("Chebyshev:", accuracy3)
print("Minkowski:", accuracy4)
```

F. Menyimpan Model dengan Akurasi Tertinggi

Terdapat satu model yang tinggi yaitu Manhattan. Dan kelompok kami memilih model Manhattan untuk keakuratannya.

```
# Melakukan pemilihan manhattan (tertinggi)
saved_model = pickle.dumps(knn1)

knn1_from_pickle = pickle.loads(saved_model)

knn1_array = knn1_from_pickle.predict(X_test)
print(len(knn1_array))

out_data = {'idData': [i for i in range(len(knn1_array) - 1)],
            'Klasifikasi': [knn1_array[i] for i in range(len(knn1_array) - 1)],
            'Akurasi': accuracy1 * 100
           }
out = pd.DataFrame(out_data, columns=['idData', 'Klasifikasi', 'Akurasi'])
out.to_excel("OutputLatih.xlsx")
print(out)
```

```
Euclidean: 0.8266666666666667
Manhattan: 0.8366666666666667
Chebyshev: 0.5033333333333333
Minkowski: 0.8266666666666667
```

BAB IV
HASIL DAN PENGUJIAN

A. Pengujian Classifier Seluruh Model

Dilakukan pengujian classifier pada seluruh model metric yang ada. Di mana parameter untuk n_neighbors=k di mana k adalah nilai yang dipilih secara random minimal terdapat 2 buah. Serta untuk metric-nya adalah perhitungan jarak yang akan digunakan.

- Euclidean

```
# Euclidean
knn_e2 = KNeighborsClassifier(n_neighbors=2, weights='uniform', algorithm='auto', leaf_size=30, metric="euclidean")
knn_e3 = KNeighborsClassifier(n_neighbors=3, weights='uniform', algorithm='auto', leaf_size=30, metric="euclidean")
knn_e4 = KNeighborsClassifier(n_neighbors=4, weights='uniform', algorithm='auto', leaf_size=30, metric="euclidean")
knn_e5 = KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, metric="euclidean")
```

```
# Euclidean
knn_e2.fit(X_train, y_train)
knn_e3.fit(X_train, y_train)
knn_e4.fit(X_train, y_train)
knn_e5.fit(X_train, y_train)
```

- Manhattan

```
# Manhattan
knn_m2 = KNeighborsClassifier(n_neighbors=2, weights='uniform', algorithm='auto', leaf_size=30, metric="manhattan")
knn_m3 = KNeighborsClassifier(n_neighbors=3, weights='uniform', algorithm='auto', leaf_size=30, metric="manhattan")
knn_m4 = KNeighborsClassifier(n_neighbors=4, weights='uniform', algorithm='auto', leaf_size=30, metric="manhattan")
knn_m5 = KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, metric="manhattan")
```

```
# Manhattan
knn_m2.fit(X_train, y_train)
knn_m3.fit(X_train, y_train)
knn_m4.fit(X_train, y_train)
knn_m5.fit(X_train, y_train)
```

- Chebyshev

```
# Chebyshev
knn_c2 = KNeighborsClassifier(n_neighbors=2, weights='uniform', algorithm='auto', leaf_size=30, metric="chebyshev")
knn_c3 = KNeighborsClassifier(n_neighbors=3, weights='uniform', algorithm='auto', leaf_size=30, metric="chebyshev")
knn_c4 = KNeighborsClassifier(n_neighbors=4, weights='uniform', algorithm='auto', leaf_size=30, metric="chebyshev")
knn_c5 = KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, metric="chebyshev")
```

```
# Chebyshev
knn_c2.fit(X_train, y_train)
knn_c3.fit(X_train, y_train)
knn_c4.fit(X_train, y_train)
knn_c5.fit(X_train, y_train)
```

- Minkowski

```
# Minkowski
knn_w2 = KNeighborsClassifier(n_neighbors=2, weights='uniform', algorithm='auto', leaf_size=30, metric="minkowski")
knn_w3 = KNeighborsClassifier(n_neighbors=3, weights='uniform', algorithm='auto', leaf_size=30, metric="minkowski")
knn_w4 = KNeighborsClassifier(n_neighbors=4, weights='uniform', algorithm='auto', leaf_size=30, metric="minkowski")
knn_w5 = KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, metric="minkowski")
```

```
# Minkowski
knn_w2.fit(X_train, y_train)
knn_w3.fit(X_train, y_train)
knn_w4.fit(X_train, y_train)
knn_w5.fit(X_train, y_train)
```

B. Hasil Pengujian Seluruh Model

Setelah mengevaluasi dan menguji didapati hasil dari seluruh model yaitu:

- Euclidean

```
Euclidean: 0.81
Euclidean: 0.8033333333333333
Euclidean: 0.8033333333333333
Euclidean: 0.8266666666666667
```

- Manhattan

```
Manhattan: 0.84
Manhattan: 0.84
Manhattan: 0.84
Manhattan: 0.84
```

- Chebyshev

```
Chebyshev: 0.5166666666666667
Chebyshev: 0.4766666666666667
Chebyshev: 0.48333333333333334
Chebyshev: 0.5033333333333333
```

- Minkowski

```
Minkowski: 0.81
Minkowski: 0.8033333333333333
Minkowski: 0.8033333333333333
Minkowski: 0.8266666666666667
```

Bisa diketahui bahwa model Manhattan akurasinya lebih tinggi daripada 3 model lainnya, oleh karena itu kami menyimpan model Manhattan sebagai model dengan akurasi tertinggi diantara yang lainnya.

C. Hasil Keluaran Sistem

- File OutputLatih.xls

	A	B	C	D	E
1		idData	Klasifikasi	Akurasi	
2	**0**	0	8	83.6667	
3	**1**	1	5	83.6667	
4	**2**	2	7	83.6667	
5	**3**	3	5	83.6667	
6	**4**	4	8	83.6667	
7	**5**	5	8	83.6667	
8	**6**	6	7	83.6667	
9	**7**	7	5	83.6667	
10	**8**	8	7	83.6667	
11	**9**	9	1	83.6667	
12	**10**	10	1	83.6667	

- File OutputSubmit.xls

	A	B	C
1		idData	Klasifikasi
2	**0**	0	0
3	**1**	1	0
4	**2**	2	0
5	**3**	3	0
6	**4**	4	0
7	**5**	5	0
8	**6**	6	0
9	**7**	7	0
10	**8**	8	0
11	**9**	9	0

BAB V
KESIMPULAN, KONTRIBUSI, DAN LAMPIRAN

A. Kesimpulan

Berdasarkan pengujian serta evaluasi terhadap sebanyak empat (4) model dan metode performansi, didapatkan hasil akurasi tertinggi dari model manhattan. Setelah mengetahui akurasi tertinggi dari model manhattan, dilakukannya analisis terhadap kasus handwriting digit recognizer.

Untuk pengembangan penelitian selanjutnya, berdasarkan hasil penelitian maka saran yang dapat disampaikan adalah sebagai berikut:

- Perlu dilakukan pengujian yang lebih mendalam seperti menganalisa pengaruh matrix pada semua model distance.
- Diharapkan penelitian selanjutnya menganalisa jumlah cluster yang paling optimal terhadap klasifikasi data yang lebih besar.
- Penelitian selanjutnya diperlukan pendekatan yang lain untuk menentukan dan memilih titik pusat cluster dan jarak matrix yang lebih baik untuk jenis data yang sama ataupun dengan jenis data yang lain.

B. Aktivitas Kontribusi

Nama	Kontribusi
Bagas Alfito Prismawan	Memperbaiki bug dan error pada source code, menyusun sebagian laporan
Kevin Antonio Fajrin	Menyusun laporan bagian Bab III B, C, D dan E
Sultan Kautsar	Menyusun laporan bag.: Abstrak; BAB: I, II, III A, III F, IV A, V A; Menyederhanakan source code.

C. Lampiran

Link video presentasi:
https://drive.google.com/file/d/1rTYrekt21ie7af8UjTrOZJqQ34Q74RdC/view?usp=sharing