ARTIFICIAL INTELLIGENCE PROGRAMMING ASSIGNMENT 03 REPORT

Report

Created to fulfill the assignment of the Artificial Intelligence course

By:

Muhammad Farhan Akbar	1301192246
Andi Achmad Adjie	1301194264
Muhammad Furqon Fahlevi	1301194214

INFORMATICS MAJOR
SCHOOL OF COMPUTING
TELKOM UNIVERSITY
BANDUNG
2021

1. INTRODUCTION

K-Nearest Neighbours (KNN) is a lazy learning, non-parametric algorithm. It uses data with several classes to predict the classification of the new sample point.
In simple definition K-Nearest Neighbours is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure.

2. OBSERVATION ANALYSIS

We have to build a K-Nearest Neighbours based car recommendation system to choose the best 3 cars according to user input.

2.1. DETERMINING THE DISTANCE USED

According to the assignment we required to use 4 formulas to calculate the distance, namely:

- Euclidean Distance

$$d_1(x1, x2) = \sqrt{\sum_p (x1_p - x2_p)^2}$$

- Manhattan Distance

$$d_1(x1, x2) = \sum_p |x1_p - x2_p|$$

- Minkowski Distance

$$d_1(x1, x2) = \sqrt[h]{\sum_p |x1_p - x2_p|^h}$$

- Supremum Distance

$$d_1(x1, x2) = \max_{p,f} |x1_f - x2_f|$$

2.2. PRE-PROCESSING DATA TECHNIQUE

In pre-processing data, we create an empty array named '*car_arr*'. After that we read the '*Nama Mobil*' column in '*mobil*.xlsx' data and append it to an empty array, because we don't need that column to measure the distance. After that we create a new data named '*newdata*' that drops '*Nama Mobil*' column.

```
print(data.isna().sum())
car_arr = []

for i in range(len(data)):
  car_arr.append(data.iloc[i]['Nama Mobil'])

newdata = data.drop(columns=['Nama Mobil'])
```

2.3. STRATEGY ALGORITHM FOR KNN

The strategy that we used for solving K-Nearest Neighbours is we use those 4 formulas to calculate the distance to get the best 3 cars according to user input and we compare the result from those formulas.

2.3.1. EUCLIDEAN DISTANCE

In Euclidean geometry, the Euclidean distance is the usual distance between two points p and q. This distance is measured as a line segment. The Pythagorean theorem can be used to calculate this distance.

We apply the formula from above into the code:

```
def euclidDistance(x, y):
  return sqrt((x['Ukuran'] - y['Ukuran'])**2 + (x['Kenyamanan'] - y['Kenyamanan'])**2 +
              (x['Irit'] - y['Irit'])**2 + (x['Kecepatan'] - y['Kecepatan'])**2 + (x['Harga (Ratus Juta)'] - y['Harga (Ratus Juta)'])**2)
```

2.3.2. MANHATTAN DISTANCE

The Manhattan distance is the simple sum of the horizontal and vertical components, whereas the diagonal distance might be computed by applying the Pythagorean theorem.

We apply the formula from above into the code:

```
def manhattanDistance(x, y):
  return (abs(x['Ukuran'] - y['Ukuran']) + abs(x['Kenyamanan'] - y['Kenyamanan']) + abs(x['Irit'] - y['Irit']) +
          abs(x['Kecepatan'] - y['Kecepatan']) + abs(x['Harga (Ratus Juta)'] - y['Harga (Ratus Juta)']))
```

2.3.3. MINKOWSKI DISTANCE

Minkowski distance is a distance/ similarity measurement between two points in the normed vector space (N dimensional real space) and is a generalization of the Euclidean distance and the Manhattan distance.

We apply the formula from above into the code:

```
def minkowskiDistance(x, y, h):
  return (abs(x['Ukuran'] - y['Ukuran']) + abs(x['Kenyamanan'] - y['Kenyamanan']) + abs(x['Irit'] - y['Irit']) +
          abs(x['Kecepatan'] - y['Kecepatan']) + abs(x['Harga (Ratus Juta)'] - y['Harga (Ratus Juta)']))**1/h
```

2.3.4. SUPREMUM DISTANCE

Chebyshev distance is also called Maximum value distance. It examines the *absolute magnitude of the differences* between coordinates of a pair of objects. This distance can be used for both ordinal and quantitative variables.

We apply the formula from above into the code:

```
def supremumDistance(x, y):
  return max(abs(x['Ukuran'] - y['Ukuran']), abs(x['Kenyamanan'] - y['Kenyamanan']), abs(x['Irit'] - y['Irit']),
          abs(x['Kecepatan'] - y['Kecepatan']), abs(x['Harga (Ratus Juta)'] - y['Harga (Ratus Juta)']))
```

3. PROCESS BUILT

3.1. IMPORTING PANDAS AND MATH

We imported pandas for data handling because we need to import the excel to the python google colab. We import math to calculate the formula.

```
import pandas as pd
from math import sqrt
```

3.2. IMPORTING '*MOBIL.XLSX*' AND '*TEST.XLSX*'

First of all, we input or read the data of '*mobil.xlsx*' using pandas.

```
data = pd.read_excel('https://github.com/farhanakbar8/k-nearest-neighbour/blob/main/mobil.xlsx?raw=true')
data
```

	Nama Mobil	Ukuran	Kenyamanan	Irit	Kecepatan	Harga (Ratus Juta)
0	Toyota Agya	4	4	9	6	1.00
1	Daihatsu Alya	4	3	9	6	1.10
2	Toyota Avanza	6	5	6	6	2.00
3	Daihatsu Xenia	6	4	6	6	1.75
4	Xpander	7	7	6	7	2.25
5	Livina	7	7	6	7	2.10
6	Karimun	3	4	10	5	1.20
7	Toyota Innova	8	8	5	7	4.00
8	Alphard	9	10	4	8	10.00
9	Toyota Vios	5	7	9	8	2.50
10	Honda City	5	8	7	8	2.70
11	Toyota Hiace	10	5	8	6	5.00
12	Toyota Fortuner	9	8	5	8	5.00
13	Toyota Foxy	9	9	5	7	5.50
14	Toyota Corolla Altis	5	9	7	9	6.00
15	Suzuki Ertiga	7	7	7	7	2.30
16	Suzuki Carry	7	3	9	5	0.80

After we input or read the first data, we read the '*test.xlsx*' as a data test, because we want to compare between both datas.

```
data_test = pd.read_excel('https://github.com/farhanakbar8/k-nearest-neighbour/blob/main/test.xlsx?raw=true')
data_test
```

	Ukuran	Kenyamanan	Irit	Kecepatan	Harga (Ratus Juta)
0	10	4	10	7	1

3.3. COMPARE DATA

We created an empty array for the recommend data. Then, we compared '*mobil.xlsx*' and '*test.xlsx*' using all the formulas. After that, we sort the result of the recommended data.

```
k = 3
recommend_euclid = []
recommend_manhattan = []
recommend_minkowski = []
recommend_supremum = []

for i in range(len(data)):
  current_data = newdata.iloc[i]
  recommend_euclid.append([euclidDistance(current_data, test1), car_arr[i]])
  recommend_manhattan.append([manhattanDistance(current_data, test1), car_arr[i]])
  recommend_minkowski.append([minkowskiDistance(current_data, test1, 4), car_arr[i]])
  recommend_supremum.append([supremumDistance(current_data, test1), car_arr[i]])

recommend_euclid.sort()
recommend_manhattan.sort()
recommend_minkowski.sort()
recommend_supremum.sort()
```

4. RESULT

4.1. EUCLIDEAN RESULT

As you can see, the Euclidean distance formula produces the three best cars based on our file 'test.xlsx', namely the Suzuki Carry, Toyota Hiace, and Suzuki Ertiga. Suzuki carry is the result that is closest to what the user wants.

```
result_euclid = recommend_euclid[:k]
for i in result_euclid:
  print(i)
```

```
[3.878143885933063, 'Suzuki Carry']
[4.69041575982343, 'Toyota Hiace']
[5.356304696336831 5, 'Suzuki Ertiga']
```

4.2. MANHATTAN RESULT

As you can see, the manhattan distance formula produces the three best cars based on our file 'test.xlsx', namely Suzuki Carry, Toyota Agya, and Toyota Hiace. Suzuki carry is the result that is closest to what the user wants.

```
result_manhattan = recommend_manhattan[:k]
for i in result_manhattan:
  print(i)
```

```
[7.2, 'Suzuki Carry']
[8.0, 'Toyota Agya']
[8.0, 'Toyota Hiace']
```

4.3. MINKOWSKI RESULT

As you can see, the Minkowski distance formula produces the three best cars based on our file ‘test.xlsx’, namely the Suzuki Carry, Toyota Agya, and Toyota Hiace. Suzuki carry is the result that is closest to what the user wants.

```
result_minkowski = recommend_minkowski[:k]
for i in result_minkowski:
  print(i)
```

```
[1.8, 'Suzuki Carry']
[2.0, 'Toyota Agya']
[2.0, 'Toyota Hiace']
```

4.4. SUPREMUM RESULT

As you can see, the supremum distance formula produces the three best cars based on our file ‘test.xlsx’, namely Suzuki Carry, Suzuki Ertiga, and Daihatsu Xenia. Suzuki carry is the result that is closest to what the user wants.

```
result_supremum = recommend_supremum[:k]
for i in result_supremum:
  print(i)
```

```
[3.0, 'Suzuki Carry']
[3.0, 'Suzuki Ertiga']
[4.0, 'Daihatsu Xenia']
```

5. CONCLUSION

Based on the 'test.xlsx' file that we use, it produces the same output from the four formulas where the Suzuki Carry is the best car that is closest to the criteria we want. The difference in the final result is in the second and third results, except for the manhattan distance and the minkowski distance which have exactly the same result, that's because the Minkowski distance is a generalized form of the manhattan distance.

Presentation Video Link:
https://drive.google.com/file/d/1ZgI1Ag2zCvvGs_H1FSka75ZpuF9gEYeI/view?usp=sharing