LAPORAN OBSERVASI IMPLEMENTASI ALGORITMA K-NEAREST NEIGHBOR DALAM PENGOLAHAN DATA

Nama : Aditya Gumilar NIM : 1301184037 Kelas : IF-4205

Algoritma K-Nearest Neighbor (K-NN) adalah sebuah metode klasifikasi terhadap sekumpulan data berdasarkan pembelajaran data yang sudah terklasifikasikan sebelumya. Termasuk dalam supervised learning, dimana hasil query instance yang baru diklasifikasikan berdasarkan mayoritas kedekatan jarak dari kategori yang ada dalam K-NN. Dalam mengimplementasikan Algoritma K-NN, perlu memperhatikan Langkah-langkahnya. Yaitu :

1. Menentukan Parameter k (jumlah tetangga paling dekat)
2. Menghitung jarak Euclidean objek terhadap data training yang diberikan, kemudian urutkan secara ascending(tinggi ke rendah)
3. Mengumpulkan kategori dalam variable y berdasarkan nilai k pada klasifikasi nearest neighbor

Sumber : https://informatikalogi.com/algoritma-k-nn-k-nearest-neighbor/

Hasil Observasi :

1. **Pemilihan ukuran jarak yang digunakan**
   Untuk menghitung jarak menggunakan rumus Euclidean

```
def euclidean(x1, x2):
    return np.linalg.norm(x1-x2, axis=1)
```

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

2. **Teknik pemrosesan Data**
   Sebelumnya kita upload terlebih dahulu file yang telah di sediakan ke google colab, yaitu "Diabetes.csv" agar bisa dilakukan pemrosesan data.

```
df = pd.read_csv("Diabetes.csv")
df
```

- **Teknik Rekayasa Fitur**
  Dalam Teknik rekaya fitur dilakukan dengan menggunakan normalisasi. Tujuan dari normalisasi adalah untuk mengubah data menjadi bentuk skala dari 0-1 agar memudahkan dalam pengolahan data atau ketika melakukan perhitungan

```
df = (df-df.min()) / (df.max() - df.min())
```

  - Sebelum dilakukan normalisasi

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1

  - Sesudah dilakukan normalisasi

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	0.234415	0.483333	1.0
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	0.116567	0.166667	0.0
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	0.253629	0.183333	1.0

- Membuat data set Baru
  Setelah melakukan normalisasi, selanjutnya saya menggunakan slice array untuk melakukan pembagian data berupa testing set dan training set.

```
fold1_trainingset = df[['Pregnancies', 'Glucose', 'BloodPressure','SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']][0:614]
fold1_testingset = df[['Pregnancies', 'Glucose', 'BloodPressure','SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']][614:768]

fold1 = [fold1_trainingset, fold1_testingset]

fold2_trainingset = pd.concat([df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']][0:
                df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI','DiabetesPedigreeFunction', 'Age', 'Outcome']][614:768]])
fold2_testingset = df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']][461:614]

fold2 = [fold2_trainingset, fold2_testingset]

fold3_trainingset = pd.concat([df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']][0:
                df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI','DiabetesPedigreeFunction', 'Age', 'Outcome']][461:768]])
fold3_testingset = df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']][307:461]

fold3 = [fold3_trainingset, fold3_testingset]

fold4_trainingset = pd.concat([df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']][0:
                df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI','DiabetesPedigreeFunction', 'Age', 'Outcome']][307:768]])
fold4_testingset = df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']][154:307]

fold4 = [fold4_trainingset, fold4_testingset]

fold5_trainingset = df[['Pregnancies', 'Glucose', 'BloodPressure','SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']][154:768]
fold5_testingset = df[['Pregnancies', 'Glucose', 'BloodPressure','SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']][0:154]

fold5 = [fold5_trainingset, fold5_testingset]
```

3. **Strategi penggunaan algoritma K-NN**

   Sebelumnya kita kumpulkan data terlebih dahulu untuk dihitung jaraknya menggunakan rumus Euclidean dan hasil akhirnya akan disorting berdasarkan ascending, setelah itu ambil nilai dari tetangga yang paling mendekati.

```
def predict(x, data_train, k, distance_fn):
    X, y        = data_train
    dist        = distance_fn(x, X)
    nearest_idx = dist.argsort()[:k]
    y_pred      = y[nearest_idx]
    y_pred      = np.bincount(y_pred).argmax()
    return y_pred
```

```
def accuracy(y, y_pred):
    acc = (y == y_pred).mean()
    return acc
```

```
acc = []
for k in k_observe:
    y_pred = []
    for x in X_val:
        y_hat = predict(x, data_train, k, euclidean)
        y_pred.append(y_hat)
    y_pred = np.array(y_pred)

    acc_k = accuracy(y_val, y_pred)
    print('fold=%d, k=%d, acc=%0.2f' % (i, k, acc_k))
```

4. **Pemilihan nilai K terbaik untuk proses seleksi dan estimasi model K-NN**

   Dalam menentukan k terbaik dengan k_observe yaitu 1,3,5,7 dan menggunakan 5-fold cross validation, didapatkan nilai tertinggi yaitu pada k = 3

```
idx_max = np.where(acc_mean == acc_mean.max())[0][0]
print("k terbaik dari observasi dengan",k_observe,"yaitu ada pada k =",idx_max,"dan nilai akurasinya adalah : ",acc_mean.max())
```

```
k terbaik dari observasi dengan [1, 3, 5, 7] yaitu ada pada k = 3 dan nilai akurasinya adalah :  0.7538663950428657
```

```
k_observe = [1, 3, 5, 7]
# obsmin = 1
# obsmax = 50
# k_observe = range(obsmin, obsmax)
acc_fold = []
n_fold = 5

for i in range(n_fold):

    X_val = df_fold[i][1].drop("Outcome", axis=1).to_numpy()
    y_val = df_fold[i][1]["Outcome"].to_numpy().astype("int")

    X_train = df_fold[i][0].drop("Outcome", axis=1).to_numpy()
    y_train = df_fold[i][0]["Outcome"].to_numpy().astype("int")

    data_train = (X_train, y_train)

    acc = []
    for k in k_observe:
        y_pred = []
        for x in X_val:
            y_hat = predict(x, data_train, k, euclidean)
            y_pred.append(y_hat)
        y_pred = np.array(y_pred)

        acc_k = accuracy(y_val, y_pred)
        print('fold=%d, k=%d, acc=%0.2f' % (i, k, acc_k))

        acc.append(acc_k)

    acc_fold.append(acc)
```

```
print('Nilai akurasi tiap fold dengan observasi',k_observe)
i = 0
while i < len(acc_fold):
    element = acc_fold[i]
    print("Fold ke-",i,': ',acc_fold[i].mean())
    i = i + 1
print('Rerata akurasi yang didapat pada observasi',k_observe,'adalah',acc_fold.mean())
```

```
Nilai akurasi tiap fold dengan observasi [1, 3, 5, 7]
Fold ke- 0 :  0.7224025974025974
Fold ke- 1 :  0.7663398692810457
Fold ke- 2 :  0.7305194805194805
Fold ke- 3 :  0.673202614379085
Fold ke- 4 :  0.7581168831168831
Rerata akurasi yang didapat pada observasi [1, 3, 5, 7] adalah 0.7301162889398183
```

Nilai k terbaik : `0.7538663950428657`

Nilai rata-rata akurasi : `0.7301162889398183`