LAPORAN PENGANTAR KECERDASAN BUATAN

TUGAS PEMROGRAMAN 2

Disusun untuk memenuhi salah satu tugas mata kuliah Pengantar Kecerdasan Buatan

Disusun oleh kelompok 6 :

1. Herjanto Janawisuta 1301200421 / IF-44-08
2. Hilman Taris Muttaqin 1301204208 / IF-44-08

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2022

1. Deskripsi Tugas

Diberikan file traintest.xlsx yang terdiri dari dua sheet: train dan test, yang berisi dataset untuk problem klasifikasi biner (binary classification). Setiap record atau baris data dalam dataset tersebut secara umum terdiri dari nomor baris data (*id*), fitur input (*x*1 sampai *x*3), dan output kelas (*y*). Fitur input terdiri dari nilai-nilai integer dalam range tertentu untuk setiap fitur. Sedangkan output kelas bernilai biner (0 atau 1).

id	x1	x2	x3	y
1	60	64	0	1
2	54	60	11	0
3	65	62	22	0
4	34	60	0	1
5	38	69	21	0

Sheet train berisi 296 baris data, lengkap dengan target output kelas (*y*). Gunakan sheet ini untuk tahap pemodelan atau pelatihan (training) model sesuai metode yang Anda gunakan. Adapun sheet test berisi 10 baris data, dengan output kelas (*y*) yang disembunyikan. Gunakan sheet ini untuk tahap pengujian (testing) model yang sudah dilatih. Nantinya output program Anda untuk data uji ini akan dicocokkan dengan target atau kelas sesungguhnya.

2. Metode dan Proses Implementasi

- **Metode**

  Metode yang dipilih adalah KNN (K-Nearest Neighbor)

- **Membaca Data Latih/Uji**

```
traintest = files.upload()
df = pd.read_excel('./traintest.xlsx')
df = df.drop('id', axis=1)

df.head()
```

```
Choose Files  No file chosen    Upload
Saving traintest.xlsx to traintest.xlsx
```

	x1	x2	x3	y
0	60	64	0	1
1	54	60	11	0
2	65	62	22	0
3	34	60	0	1
4	38	69	21	0

- **Membagi Data**

  Memisah X dengan Y dan membagi data menjadi data training dan data test(validation).

```
x_data = df.drop('y', axis = 1).to_numpy() # ambil data x1, x2, x3
y_data = df['y'].to_numpy() # ambil data y

x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.25)

y_test
```

```
array([1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 0])
```

- **Membangun Model KNN**

  Pada model kali ini, kami menggunakan Euclidean Distance Metric. Alasannya karena Euclidean lebih cocok untuk data yang dimensi nya kecil seperti jumlah value maupun kolom. Sementara Manhattan lebih cocok untuk data yang memiliki dimensi yang sangat besar.

```
class KNN:

    def __init__(self, k): # k-closest data point
        self.k = k

    def train_data(self, x, y): # x is matriks and y is label (0, 1)
        self.x_train = x
        self.y_train = y

    def count_range(self, x_test, x_train): # using euclidean distance
        return np.sqrt(np.sum(np.subtract(x_test, x_train))**2)

    def predict(self, x):
        y_predict = []
        for i in range(len(x)):
            prediction = self._prediction(x[i])
            y_predict.append(int(prediction))

        return y_predict
```

```
def _prediction(self, x):

    # 1. count range from all training data
    count_range_arr = [self.count_range(x, train_record) for train_record in self.x_train]

    # 2. get label from y_train
    count_range_arr_with_label = np.array([[self.y_train[index], data] for index, data in enumerate(count_range_arr)]

    # 3. sorting count range array
    count_range_arr_with_label = count_range_arr_with_label[count_range_arr_with_label[:, 1].argsort()][:self.k]

    # 4. get only label
    predict_label = [i[0] for i in count_range_arr_with_label]

    # 5. get most common i
    return Counter(predict_label).most_common(1)[0][0]
```

- **Training Model**

  Di tahap ini kita bisa mengubah K sesuai keinginan hingga mendapatkan hasil yang diharapkan.

```
model = KNN(k=9)
model.train_data(x_train, y_train,)
```

- **Confusion Matrix**

```
def confussionMatriks(prediction, test):
  # TP FP
  # FN TN
  true_positive = 0
  true_negative = 0
  false_positive = 0
  false_negative = 0

  for i in range(len(test)):
    if prediction[i] == 1 and test[i] == 1:
      true_positive += 1

    if prediction[i] == 0 and test[i] == 0:
      true_negative += 1

    if prediction[i] == 1 and test[i] == 0:
      false_positive += 1

    if prediction[i] == 0 and test[i] == 1:
      false_negative += 1

  # return [[true_positive, false_positive], [false_negative, true_negative]]
  return {'true_positive': true_positive, 'true_negative': true_negative, 'false_positive': false_positive, 'false_negative': false_negative}
```

```
confussion = confussionMatriks(prediction, y_test)
print(confussion)
```

```
{'true_positive': 46, 'true_negative': 2, 'false_positive': 17, 'false_negative': 9}
```

- **Accuration dan Performance Metrics**

```
# accuracy test
def countAccuracy(y_prediction, y_test):
  correct = 0
  for i in range(len(y_prediction)):
    if y_prediction[i] == y_test[i]:
      correct += 1

  return correct / float(len(y_test)) * 100.0
```

```
def recall(confussionMatriks):
  return confussionMatriks['true_positive']/(confussionMatriks['true_positive'] + confussionMatriks['false_negative'])

def specifity(confussionMatriks):
  return confussionMatriks['true_negative']/(confussionMatriks['true_negative'] + confussionMatriks['false_negative'])

def precission(confussionMatriks):
  return confussionMatriks['true_positive']/(confussionMatriks['true_positive'] + confussionMatriks['false_positive'])

def f1Measure(precission, recall):
  return 2 * ((precission * recall) / (precission + recall))
```

```
acc = countAccuracy(prediction, y_test)
print("Accuracy: ", acc, "%")
print("Error rate: ", 100 - acc, "%")

recallValue = recall(confussion)
precissionValue = precission(confussion)
f1ScoreValue = f1Measure(precissionValue, recallValue)

print('Recall: ', recallValue*100, "%")
print('Precission: ', precissionValue*100, "%")
print('F1 Score: ', f1ScoreValue*100, "%")
```

```
Accuracy:  64.86486486486487 %
Error rate:  35.13513513513513 %
Recall:  83.63636363636363 %
Precission:  73.01587301587301 %
F1 Score:  77.96610169491525 %
```

- **Melatih Model Dengan Data Validation**

```
# looping KNN with k 1 - 10
for j in range(1, 11):
  x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.25)
  model = KNN(k=5)
  model.train_data(x_train, y_train)
  prediction = model.predict(x_test)

  print("K =", j)
  print("Accuracy: ", countAccuracy(prediction, y_test))
  print("============================================")
```

```
K = 1
Accuracy:  72.97297297297297
============================================
K = 2
Accuracy:  66.21621621621621
============================================
K = 3
Accuracy:  66.21621621621621
============================================
K = 4
Accuracy:  74.32432432432432
============================================
K = 5
Accuracy:  70.27027027027027
============================================
K = 6
Accuracy:  70.27027027027027
============================================
K = 7
Accuracy:  71.62162162162163
============================================
K = 8
Accuracy:  58.108108108108105
============================================
K = 9
Accuracy:  64.86486486486487
============================================
K = 10
Accuracy:  64.86486486486487
============================================
```

- **Menguji Model Pada Data Test**

```
model_test = KNN(k=9)
model_test.train_data(x_train, y_train)
prediction = model_test.predict(x_test)

print(np.array(prediction))
print(len(prediction))
```

```
[1 1 1 1 1 1 1 1 1 1]
10
```

```
x_result = pd.DataFrame(x_test, columns=['x1', 'x2', 'x3'])
y_result = pd.DataFrame(prediction, columns=['y'])
result = pd.merge(x_result, y_result, left_index=True, right_index=True)
result
```

	x1	x2	x3	y
0	43	59	2	1
1	67	66	0	1
2	58	60	3	1
3	49	63	3	1
4	45	60	0	1
5	54	58	1	1
6	56	66	3	1
7	42	69	1	1
8	50	59	2	1
9	59	60	0	1

- **Menyimpan Output ke File Excel**

```
Data_result = pd.ExcelWriter('Data_result.xlsx')
result.to_excel(Data_result)
Data_result.save()
```

	A	B	C	D	E
1		**x1**	**x2**	**x3**	**y**
2	**0**	43	59	2	1
3	**1**	67	66	0	1
4	**2**	58	60	3	1
5	**3**	49	63	3	1
6	**4**	45	60	0	1
7	**5**	54	58	1	1
8	**6**	56	66	3	1
9	**7**	42	69	1	1
10	**8**	50	59	2	1
11	**9**	59	60	0	1

3. **Pembagian Tugas**
    - Hilman : Membuat model KNN, Membuat Performance Measurement, Menguji Model Training dengan data Validation, Menguji Model dengan data Test.
    - Herjanto : Membagi data Train dengan data validation, Membuat Confusion Matrix, Menyimpan output ke file, Membuat Laporan
4. **Link**
    - Google Colab https://colab.research.google.com/drive/1LAjzRebLlbeocIrSrSCTCLUvUYTsYPVH?hl=id#scrollTo=hTpt8sw6Nq15
    - Youtube https://youtu.be/0sbOKY4P1mQ