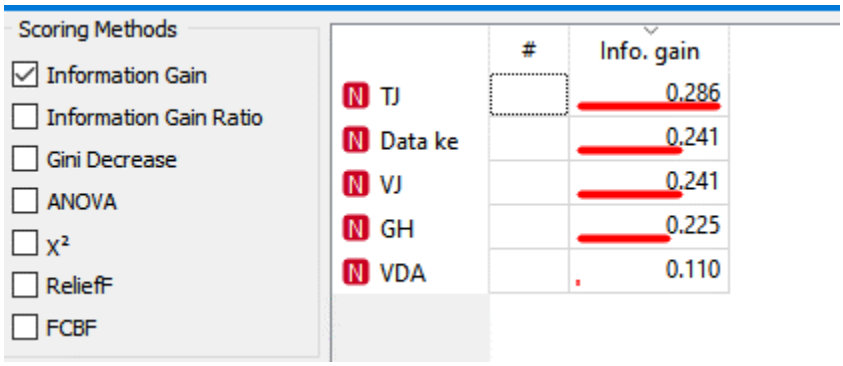
UAS Pengenalan Pola

Dyline Melynea F/185314125

Data Gunung Merapi terdapat 19 entri. Dari 19 tersebut, terdapat 6 kolom yang 1 kolomnya menjadi label yaitu kolom ‘status’. Kolom lainnya selain ‘status’ dianggap sebagai fitur kecuali kolom ‘Data ke’ karena kolom tersebut berfungsi sebagai index. Maka didapatkan 4 feature dan 1 label. Namun, dilakukan juga proses seleksi ciri untuk menentukan feature/ciri mana yang memberikan dampak.

Berikut adalah perhitungan untuk menyeleksi ciri dengan software Orange:



Selain itu juga dilakukan perhitungan manual di excel yang sebelumnya membuat kelompok/range dari masing-masing feature agar membentuk kelasnya (detail ada di file data sheet ‘f.selection’):

Hitung info gain

VJ	normal	siaga	waspada	Banyak kasus	Entropi	Gain
0-10	2	2	6	10	0.864974	
11-20	0	1	1	2	0	
31-60	1	0	1	2	0	0.544751
61-90	1	0	1	2	0	
111-160	0	1	2	3	0	

TJ	normal	siaga	waspada	Banyak kasus	Entropi	Gain
0-30	2	3	4	9	0.965634	
31-80	0	1	5	6	0	
111-140	1	0	2	3	0	0.517183

GH	normal	siaga	waspada	Banyak kasus	Entropi	Gain
0-20	3	3	6	12	0.946395	
41-70	0	0	3	3	0	
101-120	1	0	2	3	0	
151-160	0	1	0	1	0	0.402277

VDA	normal	siaga	waspada	Banyak kasus	Entropi	Gain
0-20	4	3	8	15	0.91899	
61-80	0	1	1	2	0	
111-150	0	0	2	2	0	0.274482

Dari kedua cara tersebut, sama-sama dihasilkan fitur yang layak digunakan ada 3 yaitu: ‘VJ’, ‘TJ’, dan ‘GH’

Kemudian dilakukan klasifikasi. Pertama dicoba dengan manual menggunakan excel, data dibagi menjadi 10 untuk training, dan 9 untuk testing. Digunakan KNN dengan metric/distance Manhattan dan banyak neighbor yaitu 1 untuk manual.

Didapatkan hasilnya seperti berikut (detail ada di file data sheet ‘data olahan’):

indeks	train												
	VJ	TJ	GH	Status	test1	test2	test3	test4	test5	test6	test7	test8	test9
2	2	68	41	waspada	104	86	122	192	91	88	214	74	130
3	0	35	0	waspada	152	40	112	202	57	82	172	2	60
7	144	131	53	waspada	263	303	213	219	304	211	147	291	273
9	1	13	0	waspada	173	17	135	191	34	103	193	23	41
18	52	16	7	waspada	178	62	190	130	75	156	146	76	20
6	2	15	63	waspada	107	55	197	125	60	163	253	83	99
10	17	25	157	siaga	74	174	296	130	175	262	322	182	188
16	111	79	0	siaga	231	191	179	187	204	145	75	153	137
17	1	25	109	normal	52	112	232	98	117	200	290	120	156
8	78	135	0	normal	254	214	90	224	227	96	164	176	160

test1	52	normal	waspada	F	akurasi			
test2	17	waspada	siaga	F	0.33333			
test3	90	normal	normal	T				
test4	98	normal	waspada	F				
test5	34	waspada	siaga	F				
test6	82	waspada	waspada	T				
test7	75	siaga	waspada	F				
test8	2	waspada	waspada	T				
test9	20	waspada	normal	F				

	test			
indeks	VJ	TJ	GH	Status
4	19	54	114	waspada
5	2	10	13	siaga
11	0	147	0	normal
12	85	19	101	waspada
13	4	1	19	siaga
14	2	115	0	waspada
15	157	50	0	waspada
19	2	35	0	waspada
20	39	15	1	normal

Dicoba juga ketika data baru masuk yaitu mendapatkan status waspada, dan hasilnya adalah tidak sesuai dengan kelas labelnya.

VJ	TJ	GH	Status	VJ	TJ	GH	Status	test10	pred 1-nn		
2	68	41	waspada	49	92	7	siaga	105	79	waspada	F
0	35	0	waspada					113			
144	131	53	waspada					180			
1	13	0	waspada					134			
52	16	7	waspada					79			
2	15	63	waspada					180			
17	25	157	siaga					249			
111	79	0	siaga					82			
1	25	109	normal					217			
78	135	0	normal					79			

Dicoba juga dengan program menggunakan bahasa python3. Data dibagi menjadi 50% training, dan 50% testing. Berikut adalah dokumentasi dan rekapan dari percobaan tuning parameter.

Manhhatan, neighbot = 1, akurasi= 30%

```
In [12]: knn = KNeighborsClassifier(n_neighbors = 1, metric = 'manhattan',
                                    weights = 'uniform', algorithm = 'ball_tree', leaf_size=5)
         knn.fit(x_train, y_train)

Out[12]: KNeighborsClassifier(algorithm='ball_tree', leaf_size=5, metric='manhattan',
                              metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                              weights='uniform')

In [13]: y_pred=knn.predict(x_test)

In [14]: y_pred

Out[14]: array([3, 1, 3, 3, 3, 1, 3, 2, 3, 3], dtype=int64)

In [15]: print(metrics.classification_report(y_test, y_pred))

                    precision    recall  f1-score   support

                 1       0.00      0.00      0.00         1
                 2       0.00      0.00      0.00         3
                 3       0.43      0.50      0.46         6

          accuracy                           0.30        10
         macro avg       0.14      0.17      0.15        10
      weighted avg       0.26      0.30      0.28        10
```

Manhattan, neighbors = 3, akurasi 40%

In [16]:
```
knn = KNeighborsClassifier(n_neighbors = 3, metric = 'manhattan',
                           weights = 'uniform', algorithm = 'ball_tree', leaf_size=5)
knn.fit(x_train, y_train)
```

Out[16]:
```
KNeighborsClassifier(algorithm='ball_tree', leaf_size=5, metric='manhattan',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

In [17]:
```
y_pred=knn.predict(x_test)
```

In [18]:
```
y_pred
```

Out[18]:
```
array([3, 1, 3, 3, 3, 3, 3, 1, 3, 3], dtype=int64)
```

In [19]:
```
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         3
           3       0.50      0.67      0.57         6

    accuracy                           0.40        10
   macro avg       0.17      0.22      0.19        10
weighted avg       0.30      0.40      0.34        10
```

Euclidean, neighbors = 1, akurasi = 30%

In [20]:
```
knn = KNeighborsClassifier(n_neighbors = 1, metric = 'euclidean',
                           weights = 'uniform', algorithm = 'ball_tree', leaf_size=5)
knn.fit(x_train, y_train)
```

Out[20]:
```
KNeighborsClassifier(algorithm='ball_tree', leaf_size=5, metric='euclidean',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

In [21]:
```
y_pred=knn.predict(x_test)
```

In [22]:
```
y_pred
```

Out[22]:
```
array([3, 2, 3, 3, 1, 3, 2, 3, 3], dtype=int64)
```

In [23]:
```
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         3
           3       0.43      0.50      0.46         6

    accuracy                           0.30        10
   macro avg       0.14      0.17      0.15        10
weighted avg       0.26      0.30      0.28        10
```

Euclidean, neighbors = 3, akurasi = 40%

In [24]:
```
knn = KNeighborsClassifier(n_neighbors = 3, metric = 'euclidean',
                           weights = 'uniform', algorithm = 'ball_tree', leaf_size=5)
knn.fit(x_train, y_train)
```

Out[24]:
```
KNeighborsClassifier(algorithm='ball_tree', leaf_size=5, metric='euclidean',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

In [25]:
```
y_pred=knn.predict(x_test)
```

In [26]:
```
y_pred
```

Out[26]:
```
array([3, 1, 3, 3, 3, 3, 3, 1, 3, 3], dtype=int64)
```

In [27]:
```
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         3
           3       0.50      0.67      0.57         6

    accuracy                           0.40        10
   macro avg       0.17      0.22      0.19        10
weighted avg       0.30      0.40      0.34        10
```

Digunakan maksimal banyak neighborsnya 3 karena ada 2 kelas yang memiliki entri hanya 4, dan jumlah neighbor lebih baik ganjil, sehingga dipilih 1 atau 3, dari hasil program ini maka didapatkan hasil paling baik yaitu dengan neighbors sebanyak 3 dengan akurasi 40% baik untuk metric/distance Manhattan maupun Euclidean.

Kemudian juga dilakukan Time series cross validation yaitu seperti berikut:

n_split = 3

```
In [38]: from sklearn.model_selection import TimeSeriesSplit

In [39]: tscv = TimeSeriesSplit()
         TimeSeriesSplit(max_train_size=None, n_splits=3)

Out[39]: TimeSeriesSplit(max_train_size=None, n_splits=3)

In [41]: for train_index, test_index in tscv.split(x):
             print("TRAIN (index data):", train_index, "TEST (index data):", test_index)
             X_train, X_test = x[train_index], x[test_index]
             y_train, y_test = y[train_index], y[test_index]
             knn = KNeighborsClassifier(n_neighbors = 3, metric = 'manhattan',
                                        weights = 'uniform', algorithm = 'ball_tree', leaf_size=5)
             knn.fit(X_train, y_train)
             y_pred=knn.predict(X_test)
             print("hasil prediksi:", y_pred)
             print(metrics.classification_report(y_test, y_pred),"\n")
```

```
TRAIN (index data): [0 1 2 3] TEST (index data): [4 5 6]
hasil prediksi: [3 3 3]
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           3       0.67      1.00      0.80         2

    accuracy                           0.67         3
   macro avg       0.33      0.50      0.40         3
weighted avg       0.44      0.67      0.53         3


TRAIN (index data): [0 1 2 3 4 5 6] TEST (index data): [7 8 9]
hasil prediksi: [3 3 3]
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         1
           3       0.33      1.00      0.50         1

    accuracy                           0.33         3
   macro avg       0.11      0.33      0.17         3
weighted avg       0.11      0.33      0.17         3
```

```
TRAIN (index data): [0 1 2 3 4 5 6 7 8 9] TEST (index data): [10 11 12]
hasil prediksi: [3 3 3]
              precision    recall  f1-score   support

           2       0.00      0.00      0.00         1
           3       0.67      1.00      0.80         2

    accuracy                           0.67         3
   macro avg       0.33      0.50      0.40         3
weighted avg       0.44      0.67      0.53         3


TRAIN (index data): [ 0  1  2  3  4  5  6  7  8  9 10 11 12] TEST (index data): [13 14 15]
hasil prediksi: [3 3 3]
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         1
           3       0.33      1.00      0.50         1

    accuracy                           0.33         3
   macro avg       0.11      0.33      0.17         3
weighted avg       0.11      0.33      0.17         3
```

```
TRAIN (index data): [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15] TEST (index data): [16 17 18]
hasil prediksi: [2 3 3]
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         0
           3       0.50      0.50      0.50         2

    accuracy                           0.33         3
   macro avg       0.17      0.17      0.17         3
weighted avg       0.33      0.33      0.33         3
```

Dari percobaan time series cross validation tersebut didapatkan hasil yang paling baik yaitu dengan skor 67% untuk iterasi pertama dan ketiga, detail indeks data yang dipakai ada seperti gambar mengenai hasil cross validation di atas. Untuk knn, konfigurasinya yang digunakan yaitu dengan Manhattan dan banyak neighbors = 3 karena neighbors 3 adalah yang paling baik sebelumnya, dan digunakan Manhattan karena hasil dari Euclidean maupun Manhattan adalah sama sebelumnya, sehingga dipilih Manhattan agar lebih sesuai dengan percobaan manual menggunakan excel.

```
In [16]: df2 = pd.read_excel(r'C:\Users\LENOVO\Documents\sem5\Pattern Recognition\UAS_185314125\data pred baru.xlsx')

In [17]: df2
```

Out[17]:

	VJ	TJ	GH	Status
0	49	92	7	2

```
In [22]: x_test = df2.iloc[:, 0:3].values

In [23]: x_test
Out[23]: array([[49, 92,  7]], dtype=int64)

In [24]: y_pred=knn.predict(x_test)
         y_pred
Out[24]: array([1], dtype=int64)
```

Dari program juga dicoba untuk memprediksi data baru, didapatkan hasilnya yaitu seperti berikut:

Hasil prediksi terhadap data baru yaitu ‘**1’** yang artinya statusnya normal. Hasilnya tidak sesuai dengan yang dicantumkan di kelas label.

Kesimpulan:

Dengan rangkaian percobaan di atas, didapatkan hasil akurasi maksimal yaitu 67%. Dan juga setelah dilakukan percobaan pada data baru, hasilnya baik menggunakan excel maupun program yaitu sama-sama tidak sesuai dengan kelas labelnya. Dapat dikatakan bahwa model KNN kurang cocok untuk prediksi data Gunung Merapi.