Klasifikasi		
Praktikan	**Aslab**	
Nama: xxxx Nim: xxxx	Annur Hangga Prihadi	065001800028
	Faiz Kumara	065001800003

PRAKTIKUM 8

DATA SAINS DAN ANALITIK

Topik pertemuan praktikum ke-delapan adalah mengolah data penilaian suatu menu dari kafe X menggunakan Klasifikasi untuk menentukan termasuk kelas apakah menu tersebut?

Source Code:

Klasifikasi Sederhana:
https://github.com/hanggaa/PrakDSDA/blob/main/Prak_8_Klasifikasi_Sederhana.ipynb

Klasifikasi dengan Validasi:
https://github.com/hanggaa/PrakDSDA/blob/main/Prak_8_Klasifikasi_dengan_Validasi.ipynb

Klasifikasi Decision Tree:
https://github.com/hanggaa/PrakDSDA/blob/main/Prak_8_Klasifikasi_Decision_Tree.ipynb

Latihan 1

Klasifikasi Sederhana

1. **Memasang library yang dibutuhkan**

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn.neighbors import KNeighborsClassifier
```

2. **Membaca data**

```
In [2]: dataset = pd.read_csv('C:/Users/hangg/Downloads/Random Aslab/DSDA/Material/Menu_Ohkaeri.csv',sep=";") #
        data = dataset[['Pelanggan', 'Karyawan', 'Stakeholder', 'Kelas']]
        data=data.dropna()
```

3. **Memisahkan data**

```
In [3]: train_data=data[['Pelanggan', 'Karyawan','Stakeholder']]
        train_label=data[['Kelas']]
```

4. **Menggunakan fungsi KNN Klasifikasi**

```
In [4]: kNN=KNeighborsClassifier(n_neighbors=3, weights='distance')
```

5. **Prediksi dengan menambah data baru**

```
In [5]: test_data=[[6, 4, 7]]
        kNN.fit(train_data, np.ravel(train_label))
        class_result=kNN.predict(test_data)

In [6]: print('Hasil klasifikasi = ', class_result.item())

        Hasil klasifikasi =  Sedang
```

Klasifikasi dengan Validasi

1. **Memasang library yang dibutuhkan**

```
In [1]: import pandas as pd
        from sklearn.model_selection import train_test_split
        import numpy as np
        from sklearn.neighbors import KNeighborsClassifier
```

2. **Membaca data**

```
In [2]: dataset = pd.read_csv('C:/Users/hangg/Downloads/Random Aslab/DSDA/Material/Menu_Ohkaeri.csv',sep=";") #
        data = dataset[['Pelanggan', 'Karyawan', 'Stakeholder', 'Kelas']]
```

3. **Memisahkan data**

```
In [3]: train, test = train_test_split(data, test_size=0.2)

In [4]: train_data=train[['Pelanggan','Karyawan','Stakeholder']]
        train_label=train[['Kelas']]
        test_data=test[['Pelanggan','Karyawan','Stakeholder']]
        test_label=test[['Kelas']]
```

4. **Just in case jika hasil data yang dilatih terdapat nilai null**

```
In [5]: pos_null = train_data.index[train_data.isnull().any(axis=1)].tolist()
        train_data = train_data.drop(pos_null)
        train_label = train_label.drop(pos_null)

In [6]: pos_null = test_data.index[test_data.isnull().any(axis=1)].tolist()
        test_data = test_data.drop(pos_null)
        test_label = test_label.drop(pos_null)
```

5. **Menentukan nilai minimum dan maximum data yang dilatih**

```
In [7]: newmin=0
        newmax=1
        mindata=train_data.min()
        maxdata=train_data.max()
        train_data = ((train_data-mindata) * (newmax-newmin) / (maxdata-mindata)) + newmin
        test_data = ((test_data-mindata) * (newmax-newmin) / (maxdata-mindata)) + newmin
```

6. **Menggunakan fungsi KNN Klasifikasi**

```
In [8]: kNN=KNeighborsClassifier(n_neighbors=3, weights='distance')
        kNN.fit(train_data, np.ravel(train_label))
        class_result=kNN.predict(test_data)

In [9]: print('Hasil klasifikasi\n', class_result)

        Hasil klasifikasi
         ['Sedang' 'Sedang' 'Sedang' 'Sedang' 'Rendah' 'Sedang' 'Sedang' 'Sedang']
```

7. **Mendeteksi data error yang diuji**

```
In [10]: error=test_label.loc[:]
         error['Class_Result']=class_result
         error['Output']=(error['Kelas'] == error['Class_Result'])

In [11]: print('\n\nPerbandingan dengan class label asli:\n', error)


         Perbandingan dengan class label asli:
               Kelas Class_Result  Output
         32  Sedang       Sedang    True
         24  Sedang       Sedang    True
         56  Sedang       Sedang    True
         28  Sedang       Sedang    True
         4   Sedang       Rendah   False
         76  Sedang       Sedang    True
         12  Sedang       Sedang    True
         64  Tinggi       Sedang   False
```

8. **Mengetahui rasio error**

```
In [12]: precision_ratio=kNN.score(test_data, test_label)
         error_ratio=1-precision_ratio

In [13]: print('\n\nError ratio = ', error_ratio)


          Error ratio =  0.25
```

Klasifikasi Decision Tree

1. **Memasang library yang dibutuhkan**

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn import tree
        from sklearn.tree import DecisionTreeClassifier
```

2. **Membaca data**

```
In [2]: dataset = pd.read_csv('C:/Users/hangg/Downloads/Random Aslab/DSDA/Material/Menu_Ohkaeri.csv',sep=";") #
        data = dataset[['Pelanggan', 'Karyawan', 'Stakeholder', 'Kelas']]
```

3. **Memisahkan data**

```
In [3]: train, test = train_test_split(data, test_size=0.2)

In [4]: train_data=train[['Pelanggan','Karyawan','Stakeholder']]
        train_label=train[['Kelas']]
        test_data=test[['Pelanggan','Karyawan','Stakeholder']]
        test_label=test[['Kelas']]
```

4. **Just in case jika hasil data yang dilatih terdapat nilai null**

```
In [5]: pos_null = train_data.index[train_data.isnull().any(axis=1)].tolist()
        train_data = train_data.drop(pos_null)
        train_label = train_label.drop(pos_null)

In [6]: pos_null = test_data.index[test_data.isnull().any(axis=1)].tolist()
        test_data = test_data.drop(pos_null)
        test_label = test_label.drop(pos_null)

In [7]: print('\nTest Data:\n', test_data)

        Test Data:
             Pelanggan  Karyawan  Stakeholder
        52          7       7.0         10.0
```

5. **Menggunakan fungsi Decision Tree**

```
In [8]: dtc=DecisionTreeClassifier(criterion='entropy', max_depth=3)
        dtc.fit(train_data, train_label)

Out[8]: DecisionTreeClassifier(criterion='entropy', max_depth=3)

In [9]: class_result=dtc.predict(test_data)
        print('\nClass = \n', class_result)


        Class =
         ['Sedang']

In [10]: acc=dtc.score(test_data, test_label)
         err=round((1-acc)*100, 2)
         print('\nError ratio = ', err, '%')


         Error ratio =  0.0 %
```

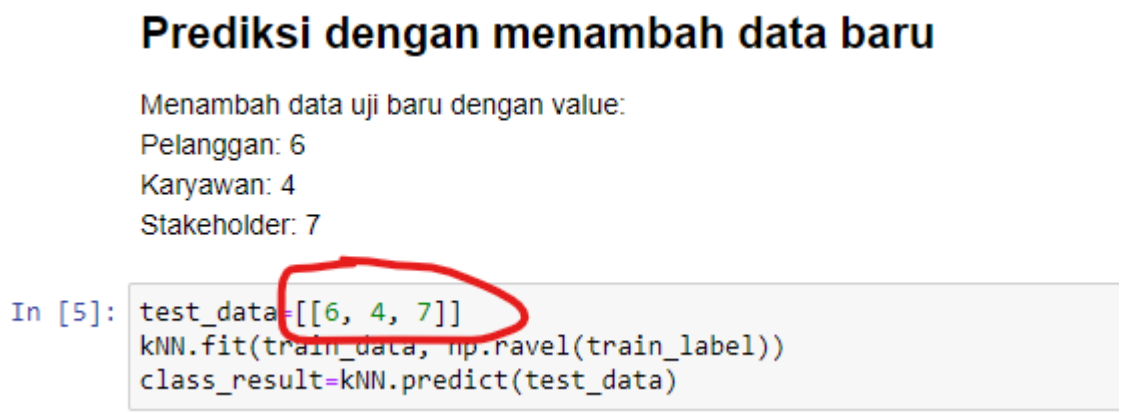
Latihan 2

1. Cari hasil prediksi kelas uji data baru dengan ketentuan berikut

Pelanggan	Karyawan	Stakeholder
6	8	2
8	8	9
8	2	4
1	4	7

Clue:

Ubah bagian yang ditandai dengan data di atas



Lampiran Screenshot hasil

Input screenshot disini

Bagaimana interpretasi hasil prediksi di atas?

Ketik interpretasi disini