Tugas Materi 5

Zul Fauzi Oktaviansyah

2110181056

3 – D4 IT - B

Dataset

```
In [2]: dataset = pd.read_csv('titanic.csv')
        dataset
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

Pengambilan dataset

Holdout Method

```
In [3]: from sklearn.model_selection import train_test_split

In [4]: label = dataset['Survived']

In [5]: X_train, X_test, Y_train, Y_test = train_test_split(dataset, label, train_size=0.7, test_size=0.3)
```

Menggunakan train_test_split dipisah antara label dengan dataset antara train = 0.7
Dan test = 0.3

Holdout Method

```
In [6]: X_train
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
283	284	1	3	Dorking, Mr. Edward Arthur	male	19.0	0	0	A/5. 10482	8.0500	NaN	S
16	17	0	3	Rice, Master. Eugene	male	2.0	4	1	382652	29.1250	NaN	Q
84	85	1	2	Ilett, Miss. Bertha	female	17.0	0	0	SO/C 14885	10.5000	NaN	S
749	750	0	3	Connaghton, Mr. Michael	male	31.0	0	0	335097	7.7500	NaN	Q
244	245	0	3	Attalah, Mr. Sleiman	male	30.0	0	0	2694	7.2250	NaN	C
...	...	...	...	...	...	...	...	...	...	...	...	...
179	180	0	3	Leonard, Mr. Lionel	male	36.0	0	0	LINE	0.0000	NaN	S
82	83	1	3	McDermott, Miss. Brigdet Delia	female	NaN	0	0	330932	7.7875	NaN	Q
603	604	0	3	Torber, Mr. Ernst William	male	44.0	0	0	364511	8.0500	NaN	S
685	686	0	2	Laroche, Mr. Joseph Philippe Lemercier	male	25.0	1	2	SC/Paris 2123	41.5792	NaN	C
259	260	1	2	Parrish, Mrs. (Lutie Davis)	female	50.0	0	1	230433	26.0000	NaN	S

623 rows × 12 columns

Hasil x_train

Holdout Method

```
In [7]: X_test
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.2250	NaN	C
587	588	1	1	Frolicher-Stehli, Mr. Maxmillian	male	60.0	1	1	13567	79.2000	B41	C
552	553	0	3	O'Brien, Mr. Timothy	male	NaN	0	0	330979	7.8292	NaN	Q
497	498	0	3	Shellard, Mr. Frederick William	male	NaN	0	0	C.A. 6212	15.1000	NaN	S
554	555	1	3	Ohman, Miss. Velin	female	22.0	0	0	347085	7.7750	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
569	570	1	3	Jonsson, Mr. Carl	male	32.0	0	0	350417	7.8542	NaN	S
708	709	1	1	Cleaver, Miss. Alice	female	22.0	0	0	113781	151.5500	NaN	S
58	59	1	2	West, Miss. Constance Mirium	female	5.0	1	2	C.A. 34651	27.7500	NaN	S
503	504	0	3	Laitinen, Miss. Kristina Sofia	female	37.0	0	0	4135	9.5875	NaN	S
215	216	1	1	Newell, Miss. Madeleine	female	31.0	1	0	35273	113.2750	D36	C

268 rows × 12 columns

Hasil x_test

Holdout Method

```
In [8]: Y_train
```

```
Out[8]: 283    1
        817    0
        451    0
        141    1
        856    1
              ..
        308    0
        712    1
        94     0
        803    1
        489    1
        Name: Survived, Length: 623, dtype: int64
```

Hasil Y_train

Holdout Method

```
In [9]: Y_test
```

```
Out[9]: 19     1
        587    1
        552    0
        497    0
        554    1
              ..
        569    1
        708    1
        58     1
        503    0
        215    1
        Name: Survived, Length: 268, dtype: int64
```

Hasil y_test

Holdout Method

```
In [10]: train_data = X_train[['Sex', 'Age', 'Pclass', 'Fare']]
         train_data
```

Out[10]:

	Sex	Age	Pclass	Fare
283	male	19.00	3	8.0500
817	male	31.00	2	37.0042
451	male	NaN	3	19.9667
141	female	22.00	3	7.7500
856	female	45.00	1	164.8667
...	...	...	...	...
308	male	30.00	2	24.0000
712	male	48.00	1	52.0000
94	male	59.00	3	7.2500
803	male	0.42	3	8.5167
489	male	9.00	3	15.9000

623 rows × 4 columns

Mengambil beberapa kolom saja dari x_train

Holdout Method

In [11]:
```
test_data = X_test[['Sex', 'Age', 'Pclass', 'Fare']]
test_data
```

Out[11]:

	Sex	Age	Pclass	Fare
19	female	NaN	3	7.2250
587	male	60.0	1	79.2000
552	male	NaN	3	7.8292
497	male	NaN	3	15.1000
554	female	22.0	3	7.7750
...	...	...	...	...
569	male	32.0	3	7.8542
708	female	22.0	1	151.5500
58	female	5.0	2	27.7500
503	female	37.0	3	9.5875
215	female	31.0	1	113.2750

268 rows × 4 columns

Mengambil beberapa kolom saja dari x_train

Holdout Method

```
In [13]: mean = train_data["Age"].mean()
         mean
Out[13]: 30.080483870967743

In [14]: train_data["Age"]= train_data["Age"].replace(np.nan, mean)
```

Mengisi nilai na dengan rata2 kolom tersebut

```
In [15]: train_data
```

Out[15]:

	Sex	Age	Pclass	Fare
283	male	19.000000	3	8.0500
817	male	31.000000	2	37.0042
451	male	30.080484	3	19.9667
141	female	22.000000	3	7.7500
856	female	45.000000	1	164.8667
...	...	...	...	...
308	male	30.000000	2	24.0000
712	male	48.000000	1	52.0000
94	male	59.000000	3	7.2500
803	male	0.420000	3	8.5167
489	male	9.000000	3	15.9000

623 rows × 4 columns

Holdout Method

```
In [16]: test_data["Age"]= test_data["Age"].replace(np.nan, 0)
```

Mengisi nilai na dengan 0

```
In [17]: test_data
```

Out[17]:

	Sex	Age	Pclass	Fare
19	female	0.0	3	7.2250
587	male	60.0	1	79.2000
552	male	0.0	3	7.8292
497	male	0.0	3	15.1000
554	female	22.0	3	7.7750
...	...	...	...	...
569	male	32.0	3	7.8542
708	female	22.0	1	151.5500
58	female	5.0	2	27.7500
503	female	37.0	3	9.5875
215	female	31.0	1	113.2750

268 rows × 4 columns

Holdout Method

```
In [18]: def min_max_scaling(data):

             data_norm = data.copy()

             for column in data_norm.columns:
                 data_norm[column] = (data_norm[column] - data_norm[column].min()) / (data_norm[column].max() - data_norm[column].min())

             return data_norm
```

```
In [19]: norm_train_data = min_max_scaling(train_data[['Age', 'Fare']])
```

```
In [20]: norm_train_data
```

```
In [21]: norm_test_data = min_max_scaling(test_data[['Age', 'Fare']])
```

```
In [22]: norm_test_data
```

Menormalisasikan train dan test data

Holdout Method

```
In [23]: from sklearn.neighbors import KNeighborsClassifier

In [24]: kNN=KNeighborsClassifier(n_neighbors=3, weights='distance')

In [25]: kNN.fit(norm_train_data, Y_train)
         class_result = kNN.predict(norm_test_data)
         class_result

Out[25]: array([1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
                0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0,
                1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1,
                0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
                0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
                0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
                0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0,
                1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0,
                1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0,
                1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0,
                0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1,
                1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0,
                1, 1, 0, 1], dtype=int64)
```

Melatih model dengan hasil normalisasi, dan yg bukan, kemudian melakukan prediksi pada Hasil normaliasi dari test data

Holdout Method

```
In [26]: precision_ratio=kNN.score(norm_test_data, Y_test)

In [27]: error_ratio=1-precision_ratio

In [28]: error_ratio
Out[28]: 0.4253731343283582
```

Melakukan perbandingan antara prediksi dan sebenarnya, kemudian Dihitung errornya

K Fold

In [29]:
```
from sklearn.model_selection import KFold
```

In [30]:
```
mean3 = dataset["Age"].mean()
dataset["Age"]= dataset["Age"].replace(np.nan, mean3)
dataset3 = min_max_scaling(dataset[['Age', 'Fare']])
```

In [31]:
```
kf=KFold(n_splits=10, shuffle=False)
print(kf)
```

```
KFold(n_splits=10, random_state=None, shuffle=False)
```

In [32]:
```
X_train = []
X_test = []
Y_train = []
Y_test = []
X = np.array(dataset[['Age', 'Fare']])
Y = np.array(label)
```

Mengisi nilai na pada kolom age dengan rata2 kolom, kemudian menginisialisasikan kfold = 10
Menggunakan library sklearn

K Fold

In [33]:
```
for train_index, test_index in kf.split(X):
    X_train.append(X[train_index])
    X_test.append(X[test_index])
    Y_train.append(Y[train_index])
    Y_test.append(Y[test_index])
```

In [34]:
```
X_train = np.array(X_train[0])
X_test = np.array(X_test[0])
Y_train = np.array(Y_train[0])
Y_test = np.array(Y_test[0])
```

In [35]:
```
kNN=KNeighborsClassifier(n_neighbors=3, weights='distance')
```

In [36]:
```
kNN.fit(X_train, Y_train)
class_result = kNN.predict(X_test)
class_result
```

Out[36]:
```
array([0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0], dtype=int64)
```

Menampung hasil kfold, dengan
Variabel baru kemudian diubah
Tipe data tersebut menjadi
Numpy array

Kemudian dilakukan KNN
Dengan K = 3 dan train data
Test data hasil dari kfold

K Fold

```
In [37]: precision_ratio=kNN.score(X_test, Y_test)

In [38]: error_ratio=1-precision_ratio

In [39]: error_ratio
Out[39]: 0.4555555555555556
```

Menghitung nilai error dari kfold

Leave One Out

```
In [40]: from sklearn.model_selection import LeaveOneOut

In [41]: loo = LeaveOneOut()

In [42]: loo.get_n_splits(X)

Out[42]: 891

In [43]: X_train2 = []
         X_test2 = []
         Y_test2 = []
         X2 = np.array(dataset[['Age', 'Fare']])
         Y2 = np.array(label)

In [44]: for train_index, test_index in loo.split(X):
             X_train2 = X2[train_index]

             temp2 = X2[test_index]
             X_test2.append(temp2[0])
             Y_train2 = Y2[train_index]

             temp = Y2[test_index]
             Y_test2.append(temp[0])
```

Menginisialisasikan leaveoneout
Dengan library sklearn
Kemudian menampung hasil dari
Leaveone out melalui variable penampung

Leave One Out

```
In [45]: X_test2 = np.array(X_test2)
         Y_test2 = np.array(Y_test2)

In [46]: kNN=KNeighborsClassifier(n_neighbors=3, weights='distance')

In [47]: kNN.fit(X_train2, Y_train2)
         class_result = kNN.predict(X_test2)
         class_result

Out[47]: array([0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
                1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
                1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1,
                0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0,
                1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0,
                0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
                0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
                0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0,
                0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
                0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
                0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
                1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1,
                0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1,
                1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0,
                0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0,
```

Mengubah tipe data menjadi Numpy array, kemudian Dilakukan proses KNN Dan prediksi

Leave One Out

```
In [48]: precision_ratio=kNN.score(X_test2, Y_test2)

In [49]: error_ratio=1-precision_ratio

In [50]: error_ratio
Out[50]: 0.050505050505050505
```

Menghitung nilai error dari leave one out