Praktikum Data Preprocessing

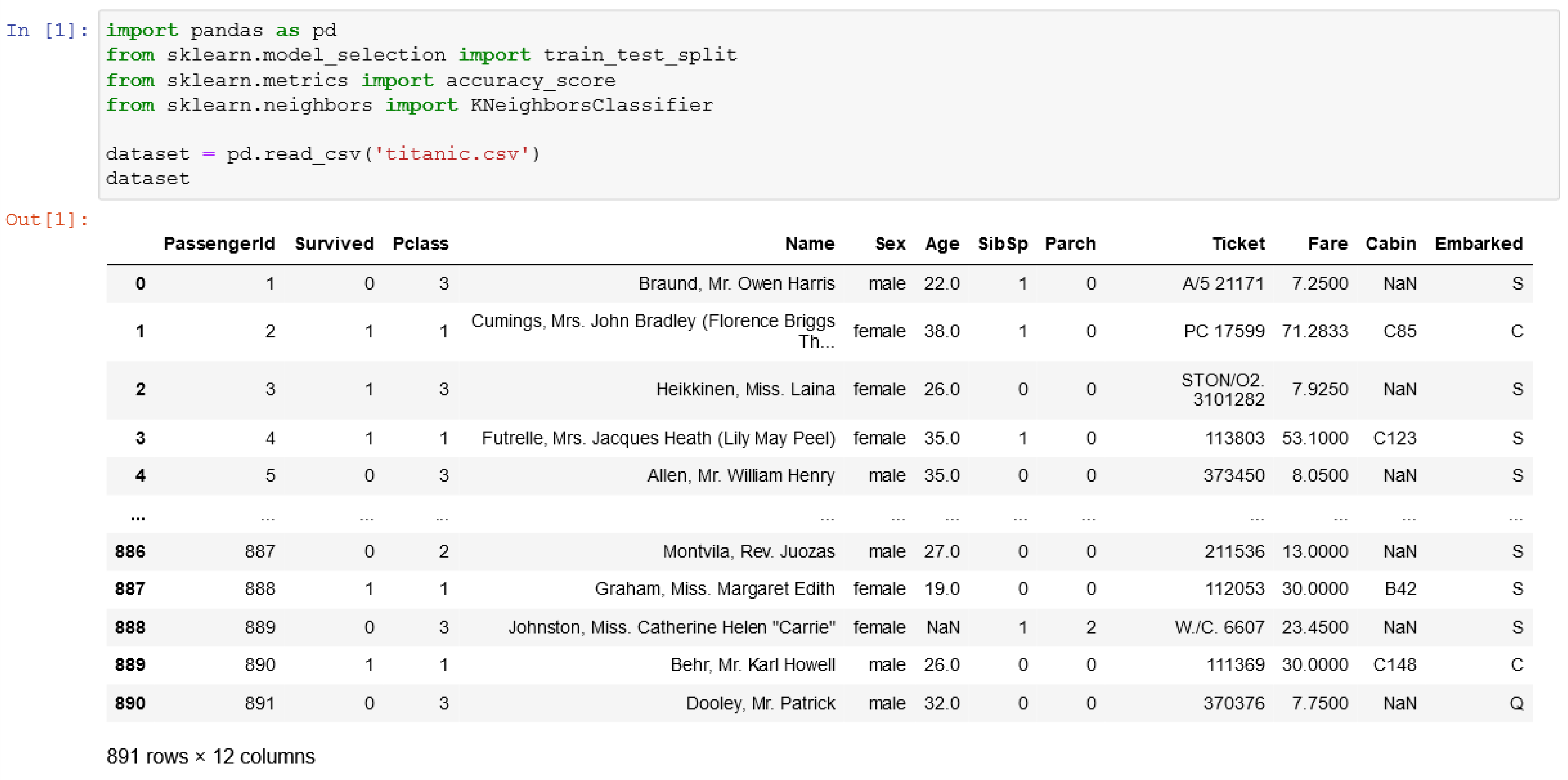
Praktikum Validation Model of Classification

Ulima Inas Shabrina(2110181048)

Assignment #5 – Validation Model

1. dataset ← titanic.csv
2. Lakukan validation Model dengan metode:
   a. Hold-out Method (70%-30%)
   b. K-Fold (k=10)
   c. LOO
3. train_data ← ambil dataset kolom fitur (Sex, Age, Pclass, Fare). Lakukan pengisian missing value pada fitur Age dengan nilai mean dari masing-masing class
4. label ← ambil dataset kolom kelas (Survived)
5. train_data ← lakukan normalisasi pada train_data dengan Min-Max 0-1 (catat nilai min dan max setiap atribut)
6. test_data ← lakukan normalisasi pada train_data dengan min-max pada Langkah 5
7. Lakukan klasifikasi k-NN (k=3) untuk masing-masing pendekatan validasi dan hitunglah error ratio-nya

1. dataset titanic.csv



In [1]:

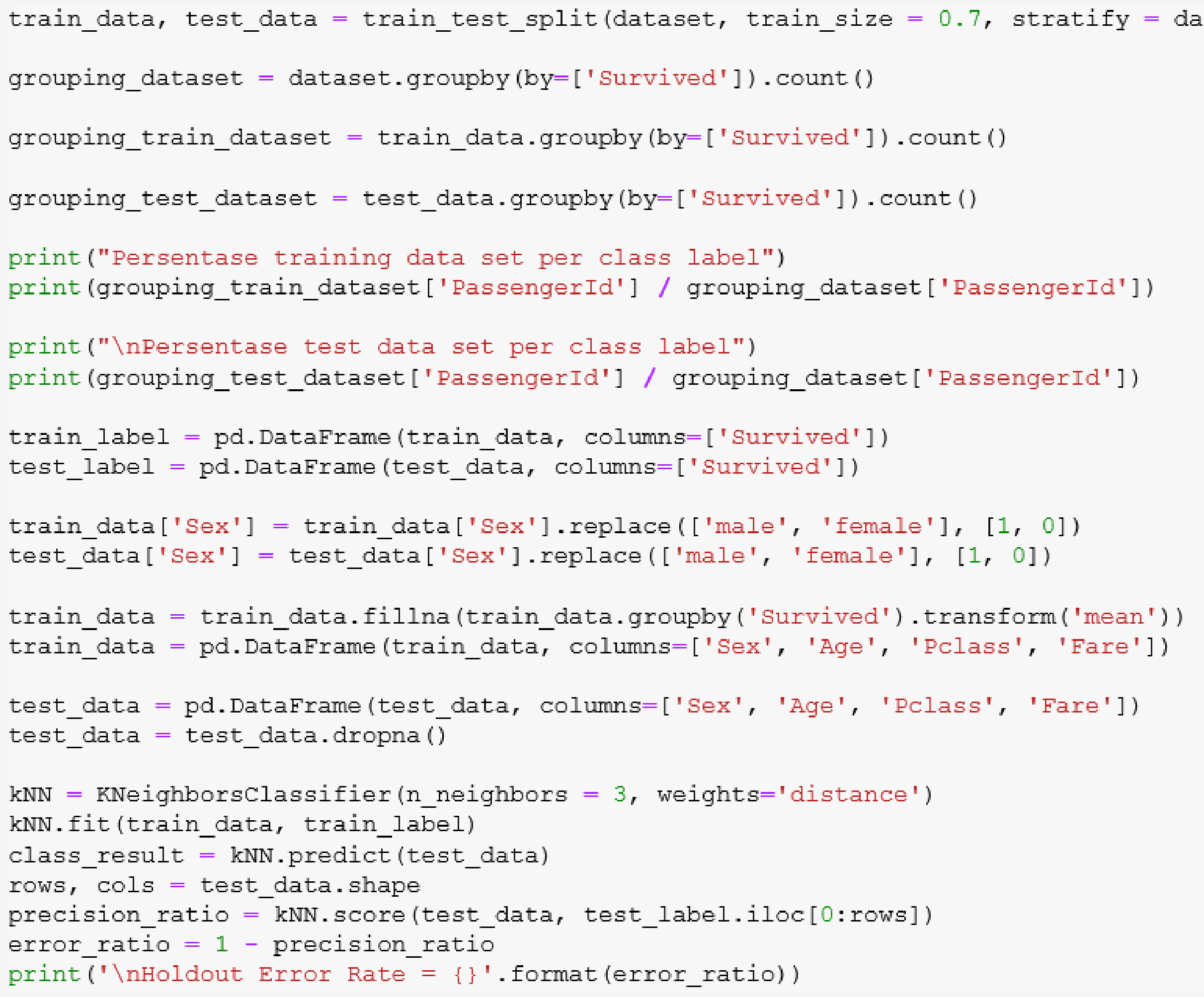
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier

dataset = pd.read_csv('titanic.csv')
dataset
```

Out[1]:

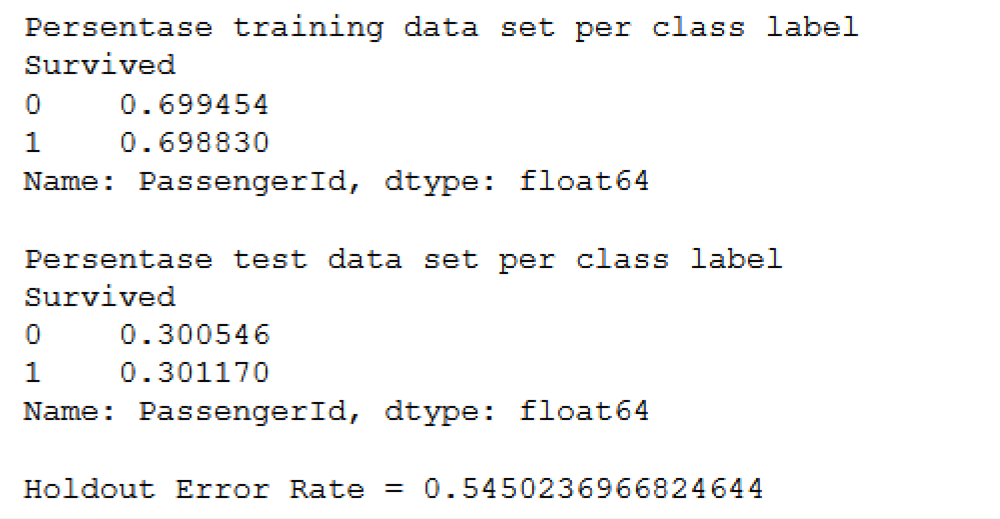
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 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

2. Validation Model Holdout

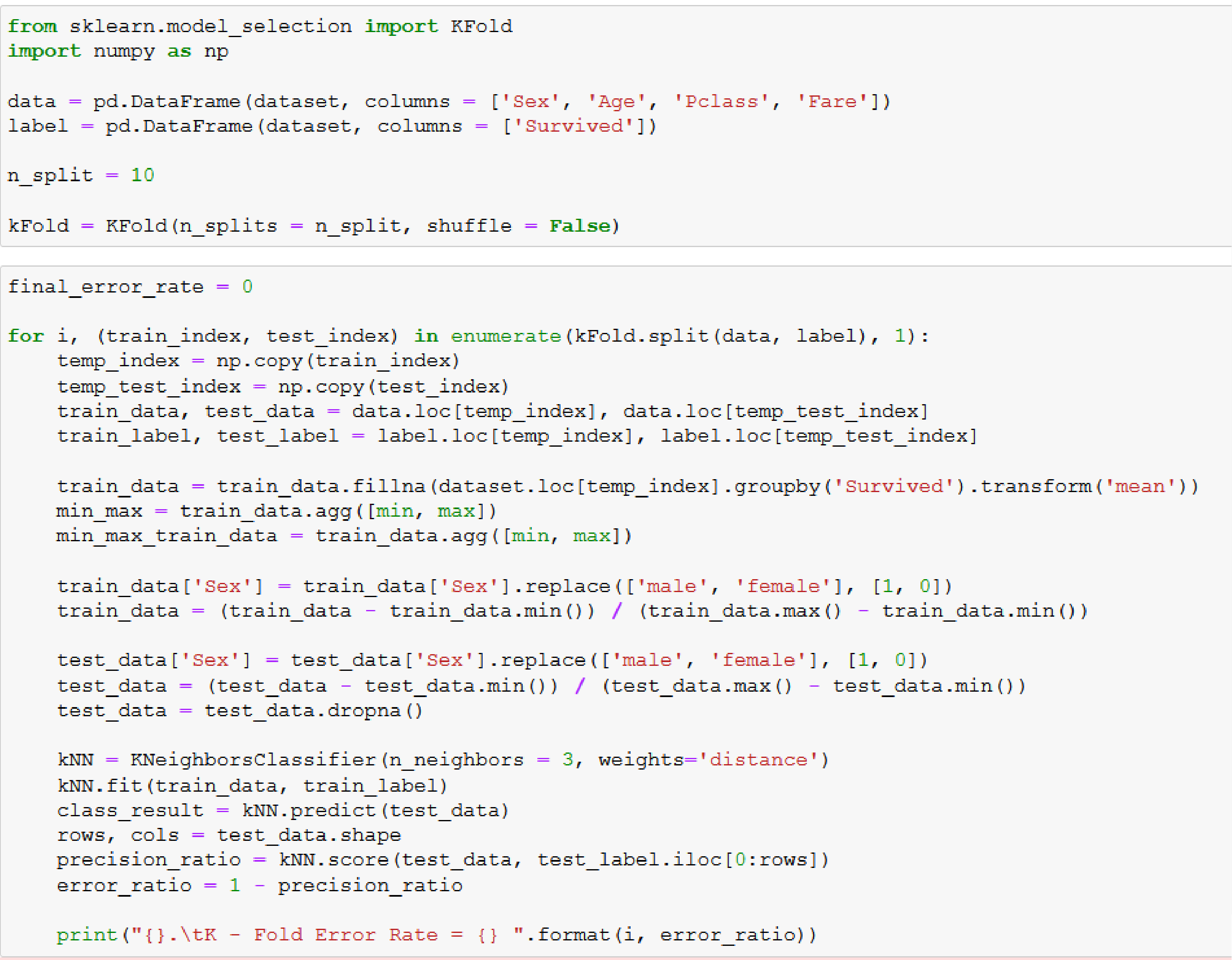


```
train_data, test_data = train_test_split(dataset, train_size = 0.7, stratify = dataset['Survived'])

grouping_dataset = dataset.groupby(by=['Survived']).count()

grouping_train_dataset = train_data.groupby(by=['Survived']).count()

grouping_test_dataset = test_data.groupby(by=['Survived']).count()

print("Persentase training data set per class label")
print(grouping_train_dataset['PassengerId'] / grouping_dataset['PassengerId'])

print("\nPersentase test data set per class label")
print(grouping_test_dataset['PassengerId'] / grouping_dataset['PassengerId'])

train_label = pd.DataFrame(train_data, columns=['Survived'])
test_label = pd.DataFrame(test_data, columns=['Survived'])

train_data['Sex'] = train_data['Sex'].replace(['male', 'female'], [1, 0])
test_data['Sex'] = test_data['Sex'].replace(['male', 'female'], [1, 0])

train_data = train_data.fillna(train_data.groupby('Survived').transform('mean'))
train_data = pd.DataFrame(train_data, columns=['Sex', 'Age', 'Pclass', 'Fare'])

test_data = pd.DataFrame(test_data, columns=['Sex', 'Age', 'Pclass', 'Fare'])
test_data = test_data.dropna()

kNN = KNeighborsClassifier(n_neighbors = 3, weights='distance')
kNN.fit(train_data, train_label)
class_result = kNN.predict(test_data)
rows, cols = test_data.shape
precision_ratio = kNN.score(test_data, test_label.iloc[0:rows])
error_ratio = 1 - precision_ratio
print('\nHoldout Error Rate = {}'.format(error_ratio))
```



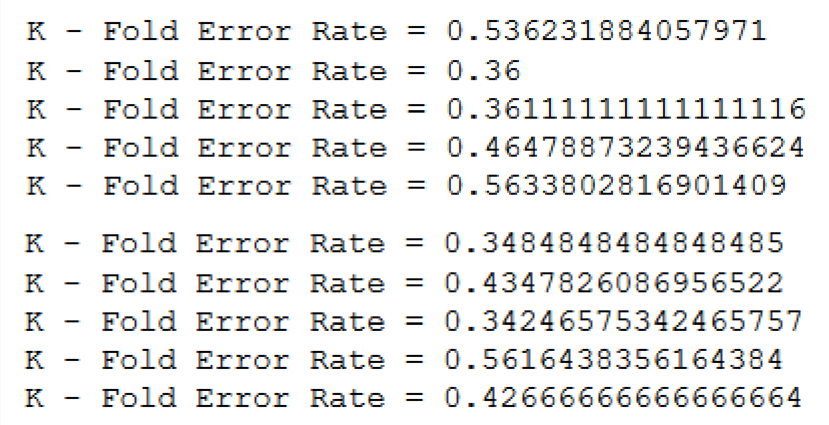
```
Persentase training data set per class label
Survived
0    0.699454
1    0.698830
Name: PassengerId, dtype: float64

Persentase test data set per class label
Survived
0    0.300546
1    0.301170
Name: PassengerId, dtype: float64

Holdout Error Rate = 0.5450236966824644
```

2. Validation Model KFold



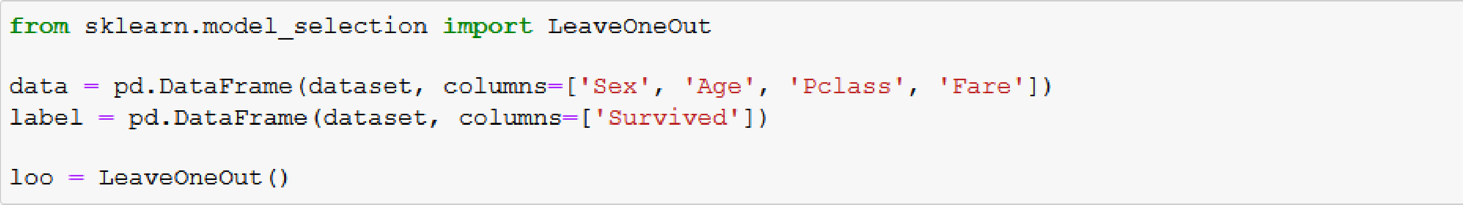
```python
from sklearn.model_selection import KFold
import numpy as np

data = pd.DataFrame(dataset, columns = ['Sex', 'Age', 'Pclass', 'Fare'])
label = pd.DataFrame(dataset, columns = ['Survived'])

n_split = 10

kFold = KFold(n_splits = n_split, shuffle = False)
```

```python
final_error_rate = 0

for i, (train_index, test_index) in enumerate(kFold.split(data, label), 1):
    temp_index = np.copy(train_index)
    temp_test_index = np.copy(test_index)
    train_data, test_data = data.loc[temp_index], data.loc[temp_test_index]
    train_label, test_label = label.loc[temp_index], label.loc[temp_test_index]

    train_data = train_data.fillna(dataset.loc[temp_index].groupby('Survived').transform('mean'))
    min_max = train_data.agg([min, max])
    min_max_train_data = train_data.agg([min, max])

    train_data['Sex'] = train_data['Sex'].replace(['male', 'female'], [1, 0])
    train_data = (train_data - train_data.min()) / (train_data.max() - train_data.min())

    test_data['Sex'] = test_data['Sex'].replace(['male', 'female'], [1, 0])
    test_data = (test_data - test_data.min()) / (test_data.max() - test_data.min())
    test_data = test_data.dropna()

    kNN = KNeighborsClassifier(n_neighbors = 3, weights='distance')
    kNN.fit(train_data, train_label)
    class_result = kNN.predict(test_data)
    rows, cols = test_data.shape
    precision_ratio = kNN.score(test_data, test_label.iloc[0:rows])
    error_ratio = 1 - precision_ratio

    print("{}.\tK - Fold Error Rate = {} ".format(i, error_ratio))
```

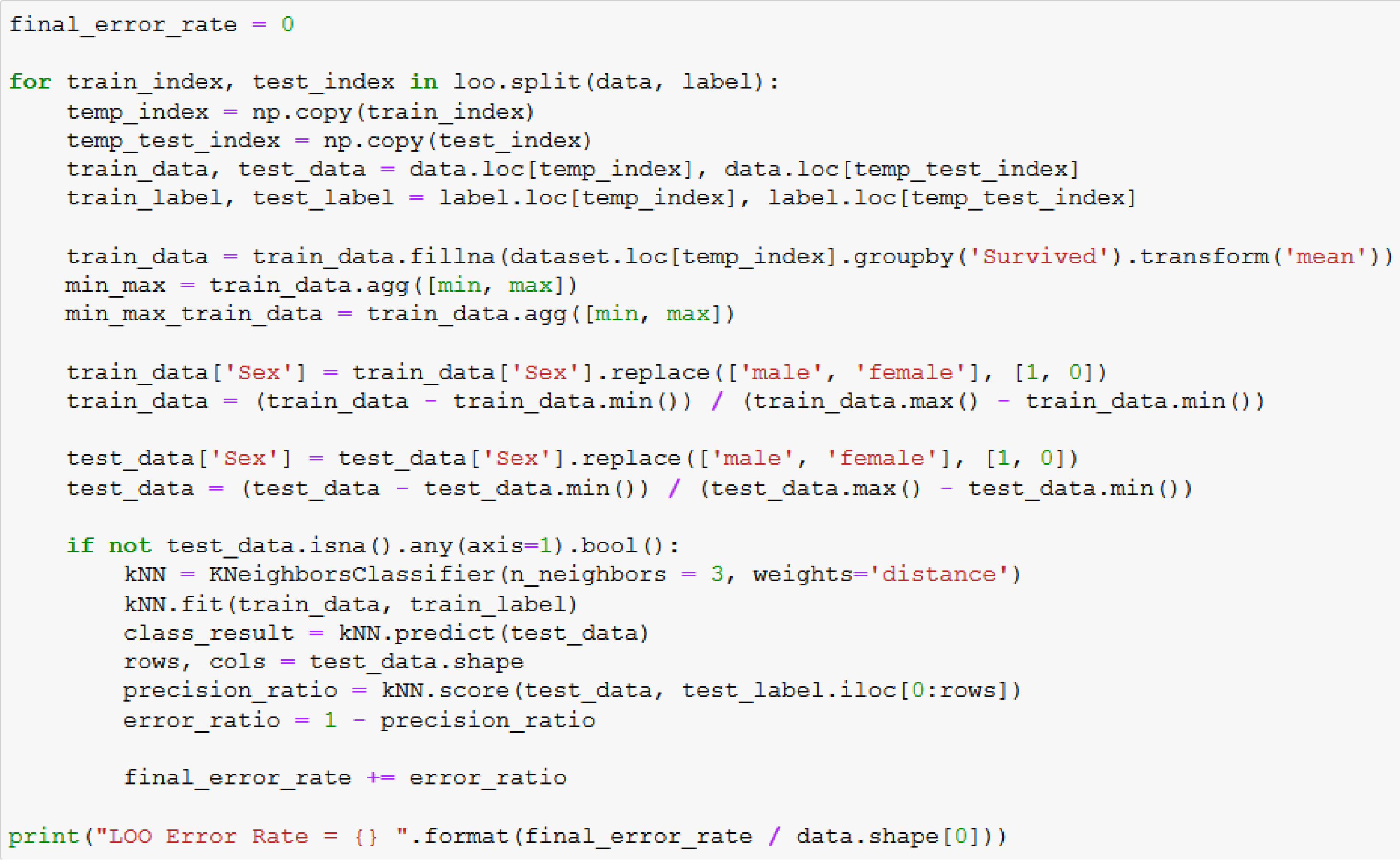


```
K - Fold Error Rate = 0.536231884057971
K - Fold Error Rate = 0.36
K - Fold Error Rate = 0.3611111111111116
K - Fold Error Rate = 0.46478873239436624
K - Fold Error Rate = 0.5633802816901409
K - Fold Error Rate = 0.3484848484848485
K - Fold Error Rate = 0.4347826086956522
K - Fold Error Rate = 0.34246575342465757
K - Fold Error Rate = 0.5616438356164384
K - Fold Error Rate = 0.42666666666666664
```

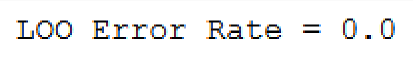
2. Validation Model LOO



```python
from sklearn.model_selection import LeaveOneOut

data = pd.DataFrame(dataset, columns=['Sex', 'Age', 'Pclass', 'Fare'])
label = pd.DataFrame(dataset, columns=['Survived'])

loo = LeaveOneOut()
```

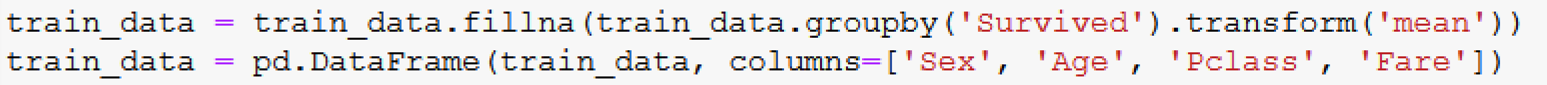


```python
final_error_rate = 0

for train_index, test_index in loo.split(data, label):
    temp_index = np.copy(train_index)
    temp_test_index = np.copy(test_index)
    train_data, test_data = data.loc[temp_index], data.loc[temp_test_index]
    train_label, test_label = label.loc[temp_index], label.loc[temp_test_index]

    train_data = train_data.fillna(dataset.loc[temp_index].groupby('Survived').transform('mean'))
    min_max = train_data.agg([min, max])
    min_max_train_data = train_data.agg([min, max])

    train_data['Sex'] = train_data['Sex'].replace(['male', 'female'], [1, 0])
    train_data = (train_data - train_data.min()) / (train_data.max() - train_data.min())

    test_data['Sex'] = test_data['Sex'].replace(['male', 'female'], [1, 0])
    test_data = (test_data - test_data.min()) / (test_data.max() - test_data.min())

    if not test_data.isna().any(axis=1).bool():
        kNN = KNeighborsClassifier(n_neighbors = 3, weights='distance')
        kNN.fit(train_data, train_label)
        class_result = kNN.predict(test_data)
        rows, cols = test_data.shape
        precision_ratio = kNN.score(test_data, test_label.iloc[0:rows])
        error_ratio = 1 - precision_ratio

        final_error_rate += error_ratio

print("LOO Error Rate = {} ".format(final_error_rate / data.shape[0]))
```

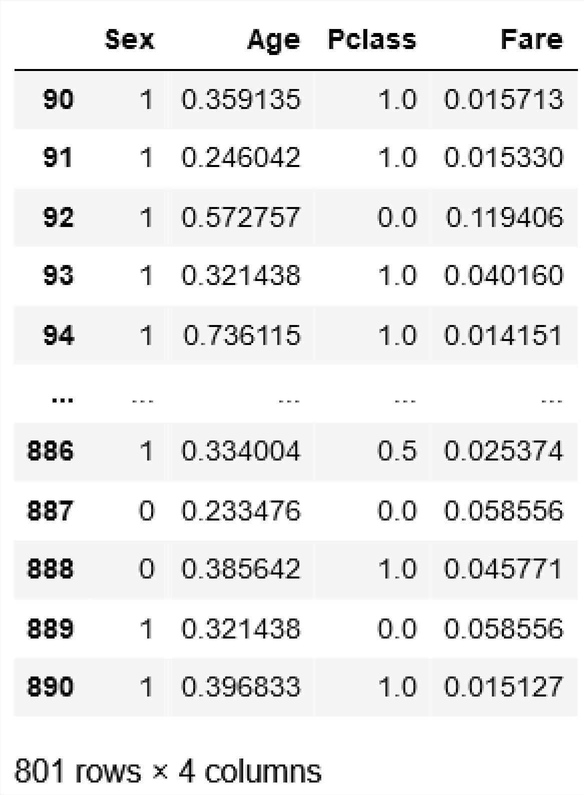


```
LOO Error Rate = 0.0
```

3. train_data -> ambil dataset kolom fitur (Sex, Age, Pclass, Fare). Lakukan pengisian missing value pada fitur Age dengan nilai mean dari masing-masing class



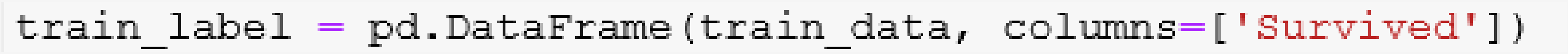
```python
train_data = train_data.fillna(train_data.groupby('Survived').transform('mean'))
train_data = pd.DataFrame(train_data, columns=['Sex', 'Age', 'Pclass', 'Fare'])
```



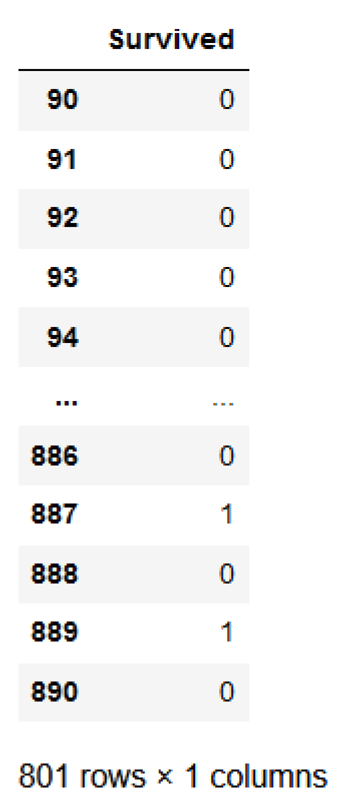
	Sex	Age	Pclass	Fare
90	1	0.359135	1.0	0.015713
91	1	0.246042	1.0	0.015330
92	1	0.572757	0.0	0.119406
93	1	0.321438	1.0	0.040160
94	1	0.736115	1.0	0.014151
...	...	...	...	...
886	1	0.334004	0.5	0.025374
887	0	0.233476	0.0	0.058556
888	0	0.385642	1.0	0.045771
889	1	0.321438	0.0	0.058556
890	1	0.396833	1.0	0.015127

801 rows × 4 columns

4. label -> ambil dataset kolom kelas (Survived)



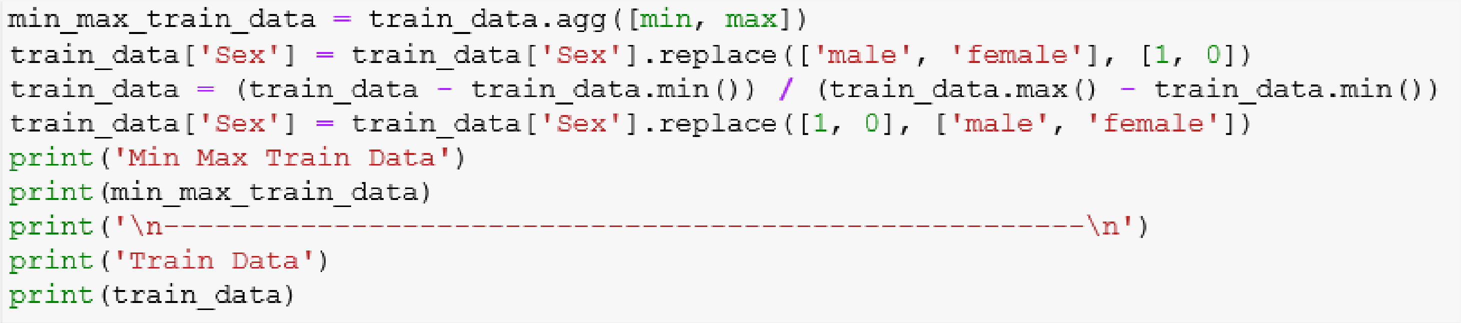
```
train_label = pd.DataFrame(train_data, columns=['Survived'])
```



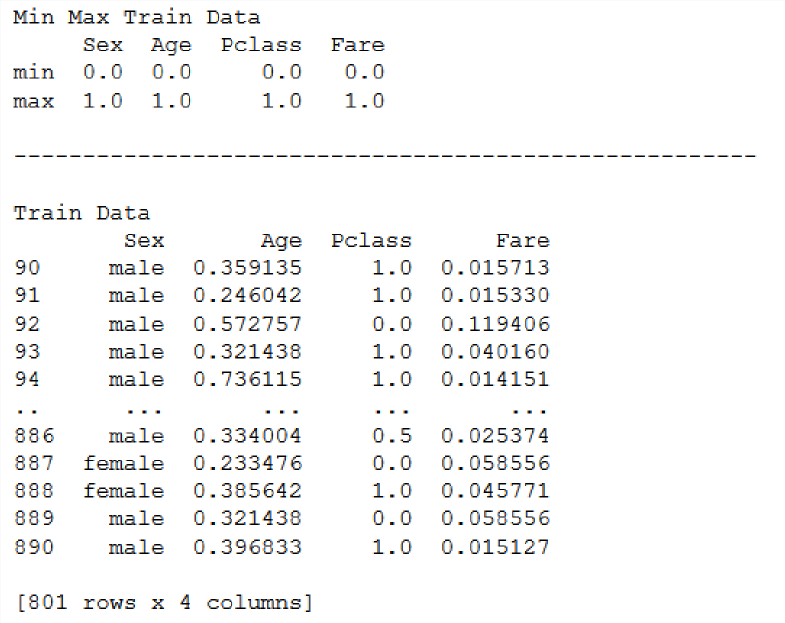
	Survived
90	0
91	0
92	0
93	0
94	0
...	...
886	0
887	1
888	0
889	1
890	0

801 rows × 1 columns

5. train_data -> lakukan normalisasi pada train_data dengan Min-Max 0-1 (catat nilai min dan max setiap atribut)

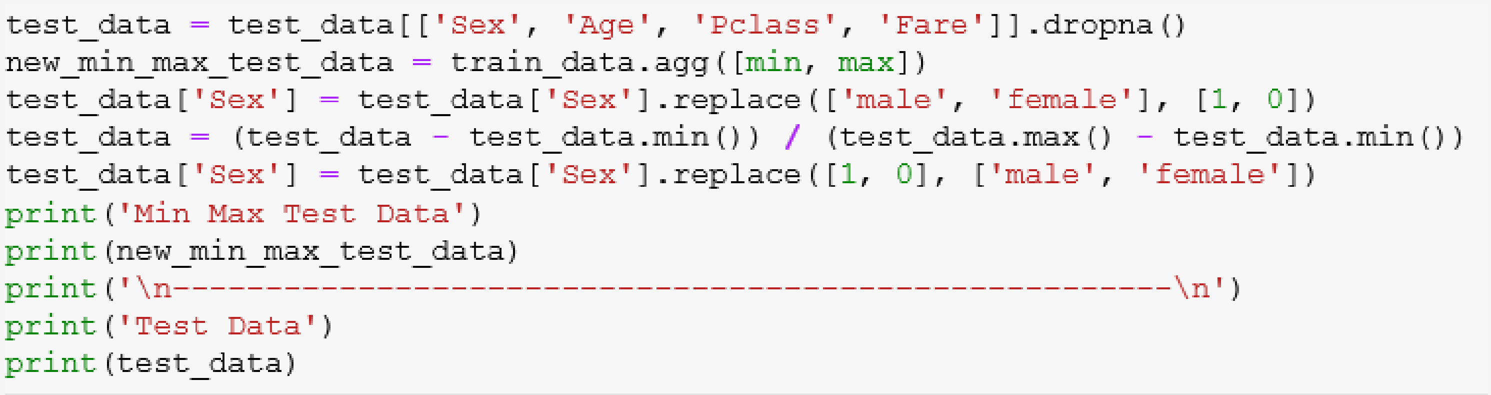


```python
min_max_train_data = train_data.agg([min, max])
train_data['Sex'] = train_data['Sex'].replace(['male', 'female'], [1, 0])
train_data = (train_data - train_data.min()) / (train_data.max() - train_data.min())
train_data['Sex'] = train_data['Sex'].replace([1, 0], ['male', 'female'])
print('Min Max Train Data')
print(min_max_train_data)
print('\n------------------------------------------------------\n')
print('Train Data')
print(train_data)
```

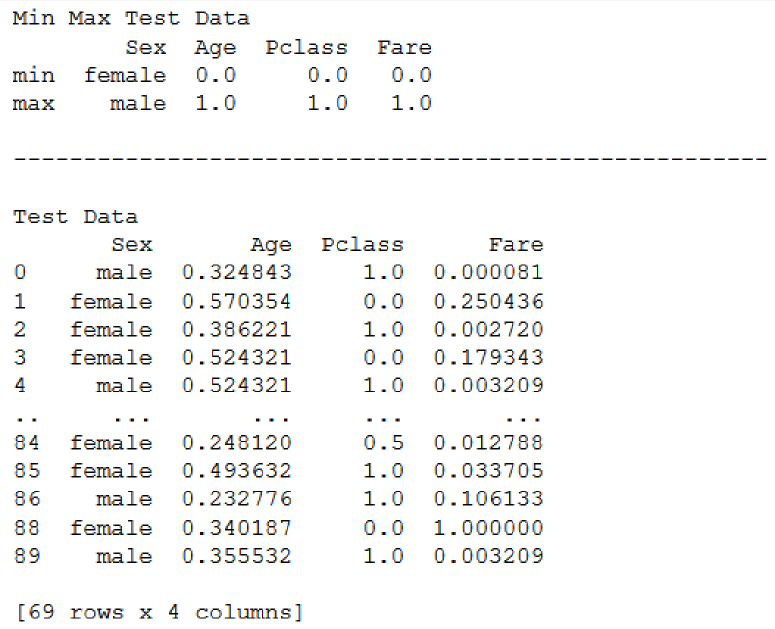


```
Min Max Train Data
     Sex  Age  Pclass  Fare
min  0.0  0.0     0.0   0.0
max  1.0  1.0     1.0   1.0

------------------------------------------------------

Train Data
        Sex       Age  Pclass      Fare
90     male  0.359135     1.0  0.015713
91     male  0.246042     1.0  0.015330
92     male  0.572757     0.0  0.119406
93     male  0.321438     1.0  0.040160
94     male  0.736115     1.0  0.014151
..      ...       ...     ...       ...
886    male  0.334004     0.5  0.025374
887  female  0.233476     0.0  0.058556
888  female  0.385642     1.0  0.045771
889    male  0.321438     0.0  0.058556
890    male  0.396833     1.0  0.015127

[801 rows x 4 columns]
```

6. test_data -> lakukan normalisasi pada train_data dengan min-max pada Langkah 5

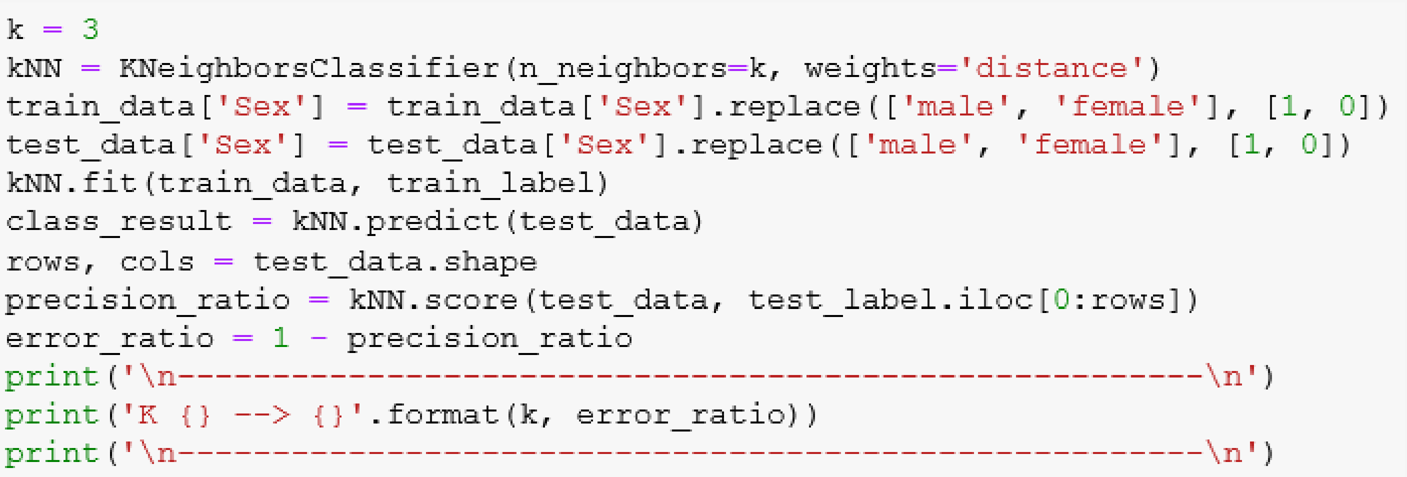


```python
test_data = test_data[['Sex', 'Age', 'Pclass', 'Fare']].dropna()
new_min_max_test_data = train_data.agg([min, max])
test_data['Sex'] = test_data['Sex'].replace(['male', 'female'], [1, 0])
test_data = (test_data - test_data.min()) / (test_data.max() - test_data.min())
test_data['Sex'] = test_data['Sex'].replace([1, 0], ['male', 'female'])
print('Min Max Test Data')
print(new_min_max_test_data)
print('\n------------------------------------------------------\n')
print('Test Data')
print(test_data)
```

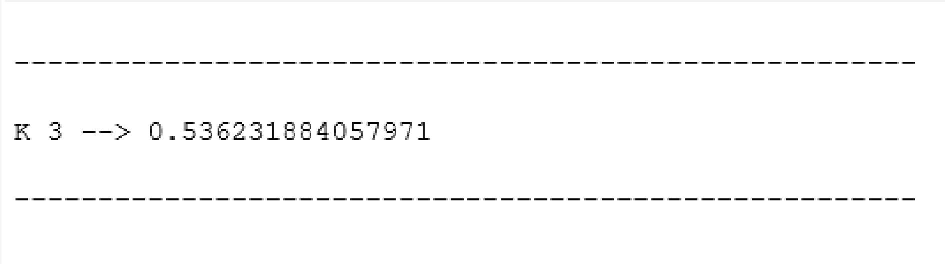


```
Min Max Test Data
        Sex  Age  Pclass  Fare
min  female  0.0     0.0   0.0
max    male  1.0     1.0   1.0

------------------------------------------------------


Test Data
       Sex       Age  Pclass      Fare
0     male  0.324843     1.0  0.000081
1   female  0.570354     0.0  0.250436
2   female  0.386221     1.0  0.002720
3   female  0.524321     0.0  0.179343
4     male  0.524321     1.0  0.003209
..     ...       ...     ...       ...
84  female  0.248120     0.5  0.012788
85  female  0.493632     1.0  0.033705
86    male  0.232776     1.0  0.106133
88  female  0.340187     0.0  1.000000
89    male  0.355532     1.0  0.003209

[69 rows x 4 columns]
```

7. Lakukan klasifikasi k-NN (k=3) untuk masing-masing pendekatan validasi dan hitunglah error ratio-nya



```python
k = 3
kNN = KNeighborsClassifier(n_neighbors=k, weights='distance')
train_data['Sex'] = train_data['Sex'].replace(['male', 'female'], [1, 0])
test_data['Sex'] = test_data['Sex'].replace(['male', 'female'], [1, 0])
kNN.fit(train_data, train_label)
class_result = kNN.predict(test_data)
rows, cols = test_data.shape
precision_ratio = kNN.score(test_data, test_label.iloc[0:rows])
error_ratio = 1 - precision_ratio
print('\n-------------------------------------------------------\n')
print('K {} --> {}'.format(k, error_ratio))
print('\n-------------------------------------------------------\n')
```



```

-------------------------------------------------------

K 3 --> 0.536231884057971

-------------------------------------------------------

```

Terimakasih