Linear regression in Sklearn

Marcin Wierzbiński

Uniwersytet Otwarty
Uniwersytetu Warszawskiego

Linear regression - Sklearn

```python
import numpy as np

from sklearn.linear_model import LinearRegression

X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])

y = np.dot(X, np.array([1, 2])) + 3

reg = LinearRegression().fit(X, y)

reg.score(X, y) #return the coefficient of determination of the prediction.

reg.coef_ #array([1., 2.])

reg.intercept_ # 3.0...

reg.predict(np.array([[3, 5]])) #array([16.])
```

model:

$$f_\theta(x) = \theta^T x$$

error:

$$\square(x, y, \theta) = (f_\theta(x) - y)^2$$

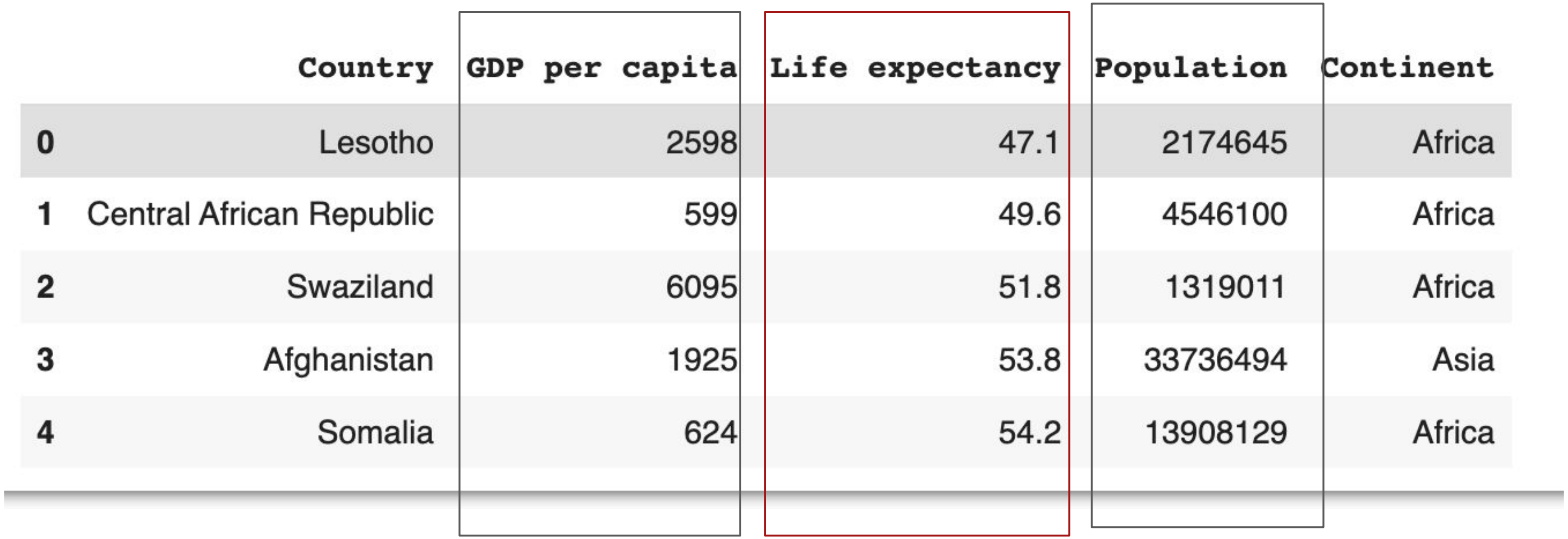
$$f_\theta(x) = 1x_0 + 2x_1 + 3$$

Coefficient of determination

```python
from sklearn.metrics import r2_score


r2 = r2_score(y_test, y_pred)
```

$$R^2 := \frac{\sum_{i=1}^{n}(\overset{\text{y\_pred}}{\overset{\downarrow}{\hat{y}_i}} - \overline{y})^2}{\sum_{i=1}^{n}(\underset{\text{y\_test}}{\underset{\uparrow}{y_i}} - \underset{\text{mean of y\_test}}{\underset{\nwarrow}{\overline{y}}})^2} \geqslant 0,$$

Model: prediction Life expectancy



	Country	GDP per capita	Life expectancy	Population	Continent
0	Lesotho	2598	47.1	2174645	Africa
1	Central African Republic	599	49.6	4546100	Africa
2	Swaziland	6095	51.8	1319011	Africa
3	Afghanistan	1925	53.8	33736494	Asia
4	Somalia	624	54.2	13908129	Africa

$$y_{life-expectancy} = \theta_2 * X_{GPD-per-capita} + \theta_1 * X_{Populaction} + \theta_0$$

Normalization and test / train split

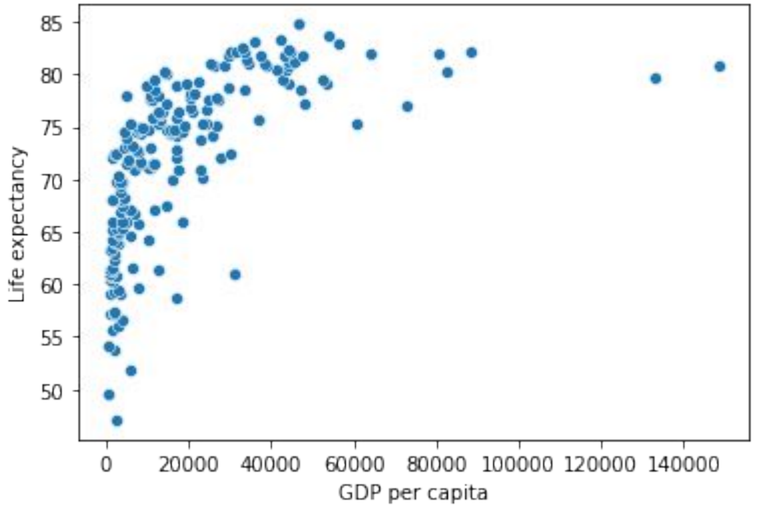
Approach:

```
import pandas as pd

data = data = pd.read_csv('income.csv', sep=';')

X = data[['Life expectancy']].to_numpy()
y = data[['GDP per capita']].to_numpy()

train_size = int(0.8 * len(X)) #80%
X_train, y_train = X[:train_size], y[:train_size]
X_test, y_test = X[train_size:], y[train_size:]

X_train = (X_train - X_train.mean()) / X_train.std()
```

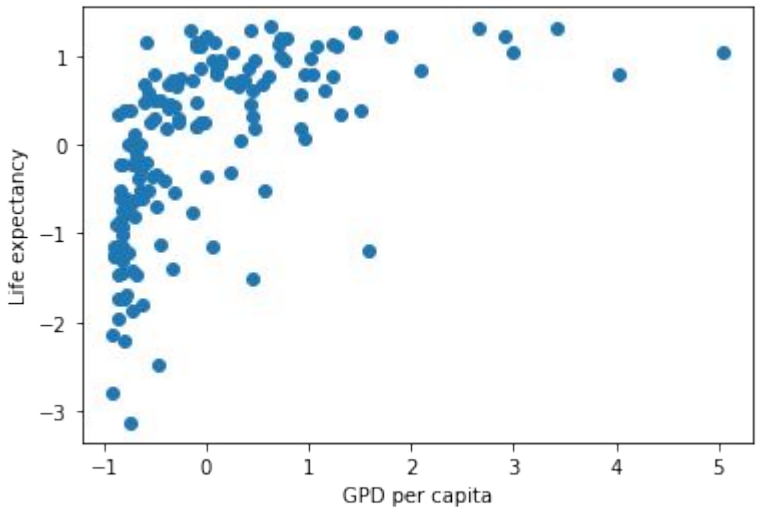
New approach:

```
from sklearn.model_selection  import train_test_split
import pandas as pd
data = pd.read_csv( 'income.csv' , sep=';')

train_data, test_data = train_test_split(data, test_size=  0.2,
random_state= 42)

X_test = test_data [['Life expectancy', 'Population']]
y_test = test_data [['GDP per capita']]
X_train = train_data [['Life expectancy', 'Population']]
y_train = train_data [['GDP per capita']]


scaler = StandardScaler()
scaler.fit(X_train)
X_train  = scaler.transform( X_train)
scaler.fit(X_test)
X_test = scaler.transform(X_test)
```

Normalization and test / train split



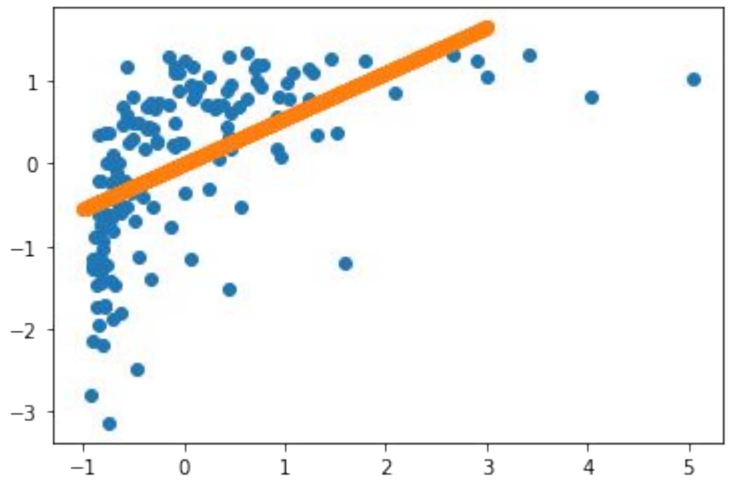
normalized



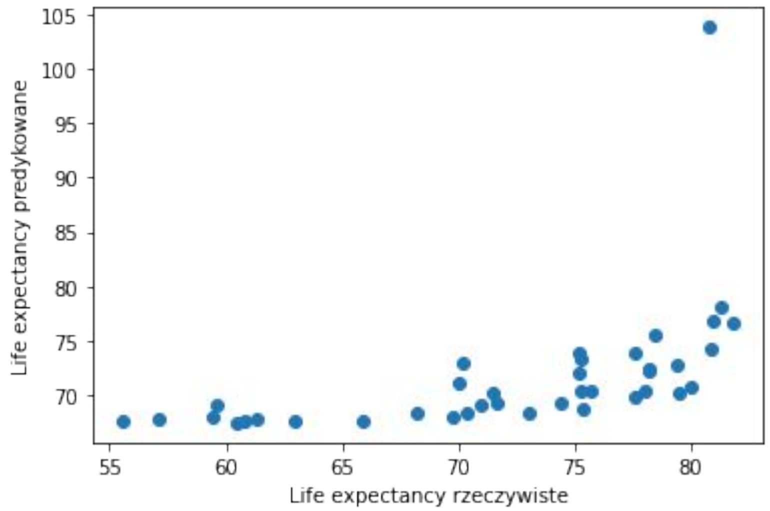
```
X = df_income[['Life expectancy']].to_numpy()
y = df_income[['GDP per capita']].to_numpy()

train_size = int(0.8 * len(X)) #80%
X_train, y_train = X[:train_size], y[:train_size]
X_test, y_test = X[train_size:], y[train_size:]
```

```
X_train = (X_train - X_train.mean()) / X_train.std()
y_train = (y_train - y_train.mean()) / y_train.std()
```

Score:



```
xs = np.linspace(-1, 3, 1000)
theta = reg.coef_
plt.scatter(y_train, X_train)
plt.scatter(xs, xs * theta[0])
```



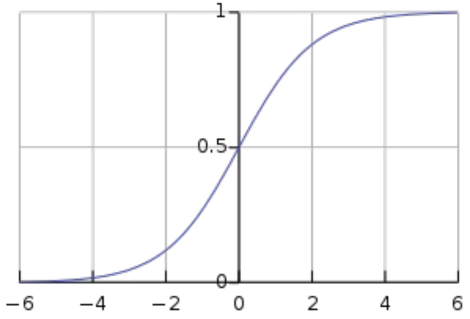
```
plt.scatter(y_test, y_pred)
plt.xlabel("Life expectancy rzeczywiste")
plt.ylabel("Life expectancy predykowane")
```

Logistic regression

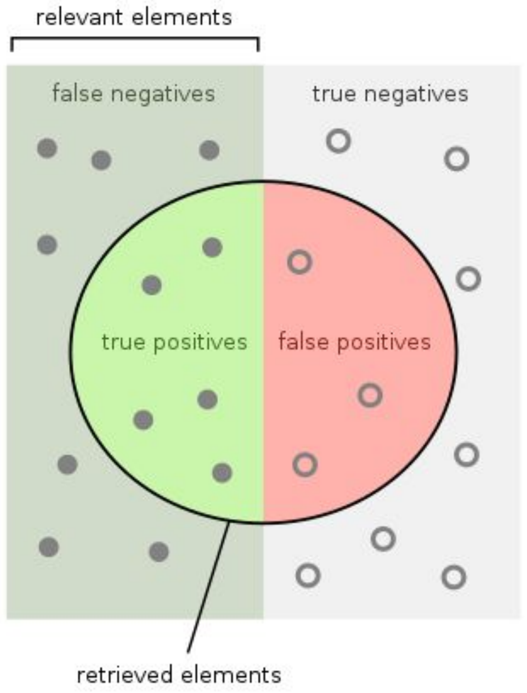
```
import numpy as np
from sklearn.linear_model import LogisticRegression
X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
y = np.array([1, 0, 0, 1])

reg = LogisticRegression().fit(X, y)
reg.score(X, y) #accuracy
reg.coef_ #array([1., 2.])
reg.intercept_ # 3.0...
reg.predict(np.array([[3, 5]])) #array([16.])
```

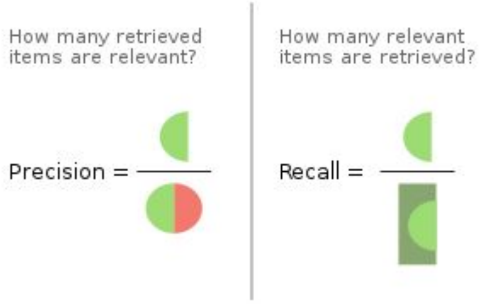
$$f_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

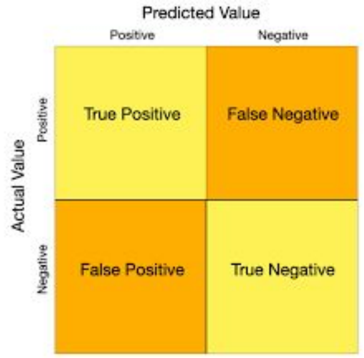
binary classification



Precision and recall




How many retrieved items are relevant?
Precision =
How many relevant items are retrieved?
Recall =

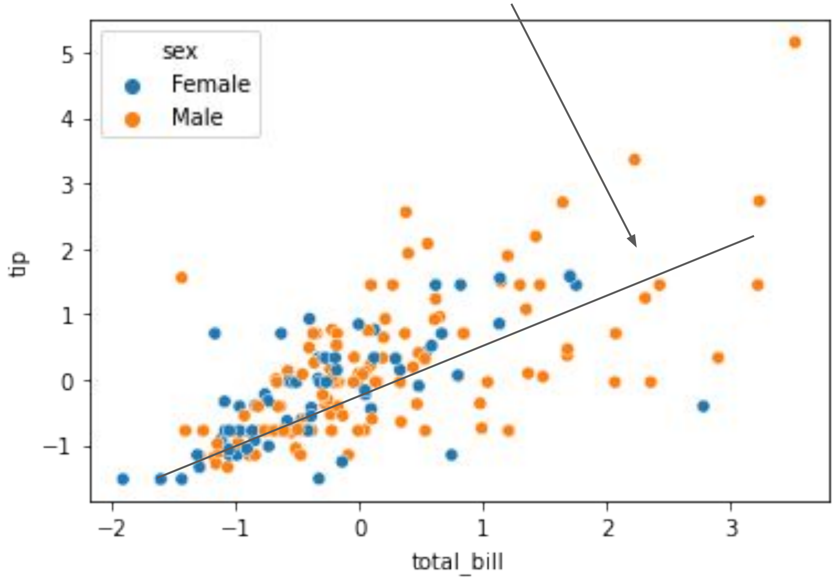


		Predicted Value	
		Positive	Negative
Actual Value	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

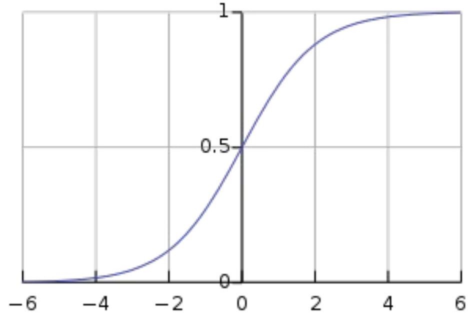
Sex prediction - bad model

decision line



```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0, max_iter=1000).fit(X_train, y_train)
clf.score(X_test, y_test) #mean accuracy
```

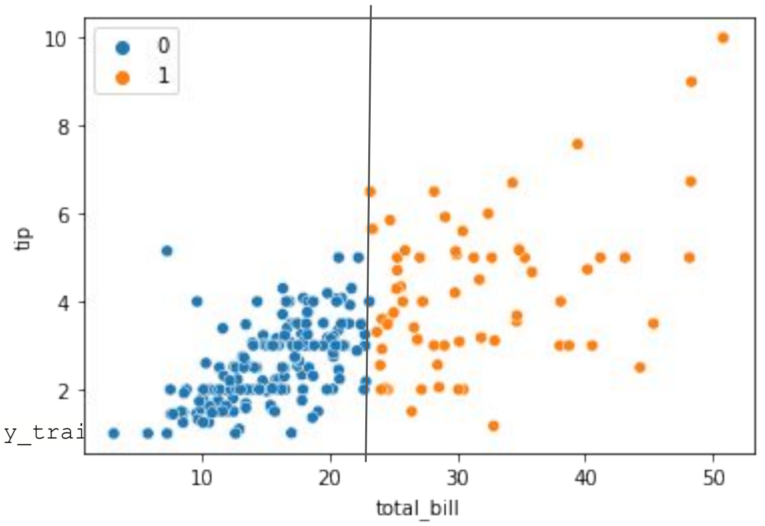
```
#accuracy 0.6122448979591837
```



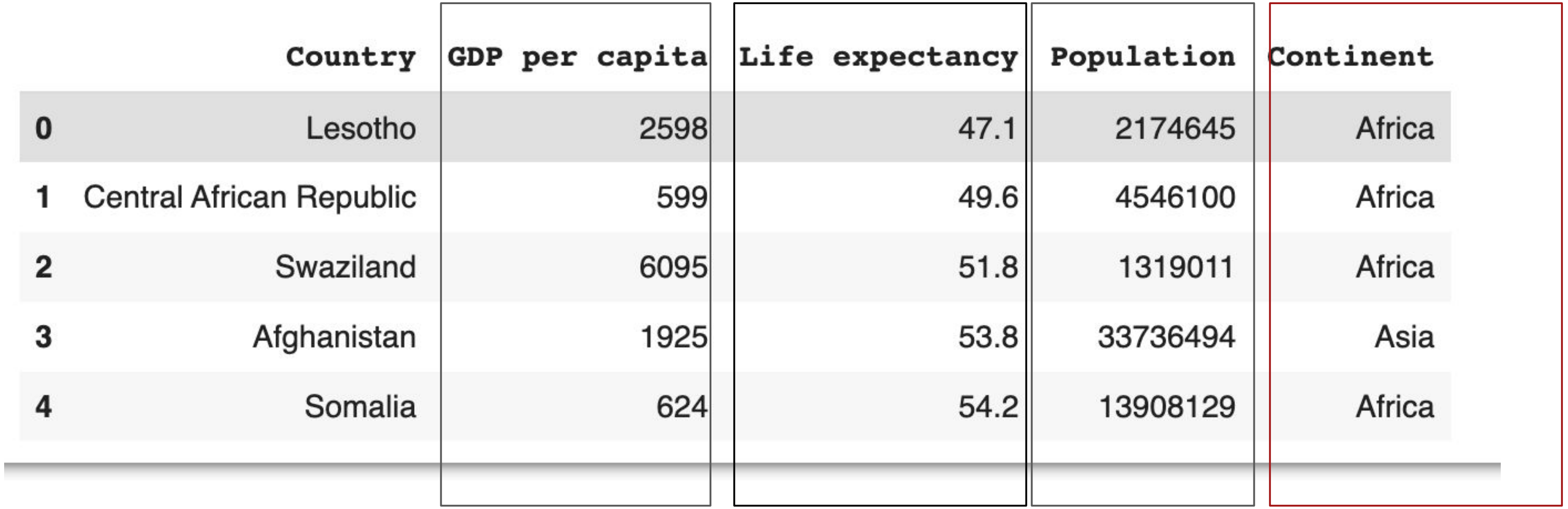
Where is the decision line?

Toy example

```
from sklearn.cluster import KMeans
from sklearn.linear_model import LogisticRegression

kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
y = kmeans.labels_
#...split data
clf = LogisticRegression(random_state=0, max_iter=100).fit(X_train, y_trai
clf.score(X_test, y_test) #1.0
y_pred = clf.predict(X_test)

# ocena jakości modelu
precision = precision_score(y_test, y_pred, average= 'macro')
recall = recall_score(y_test, y_pred, average= 'macro')
f1 = f1_score(y_test, y_pred, average= 'macro')
print("Precyzja:", precision)
print("Czułość:", recall)
print("F1-score:", f1)
```



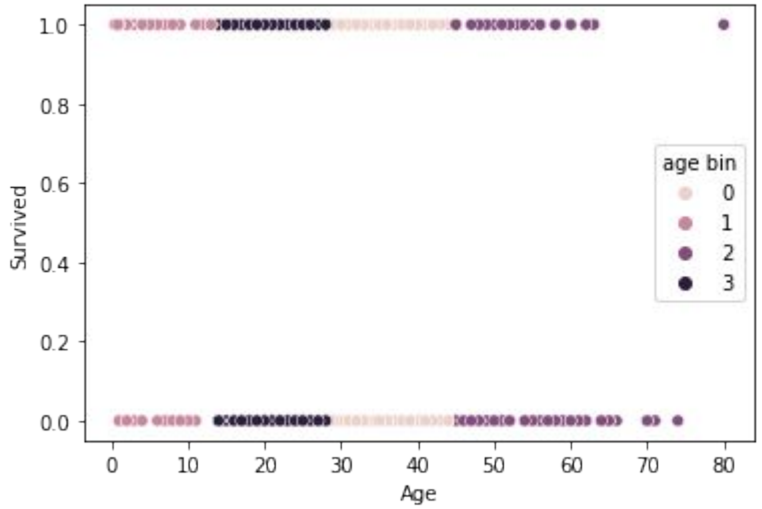
Model: prediction continent

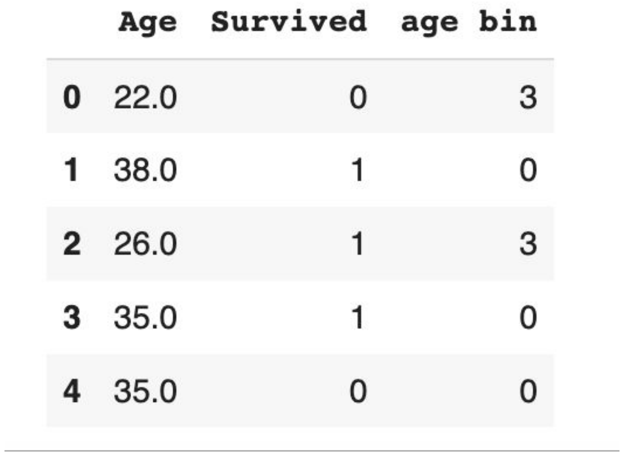


	Country	GDP per capita	Life expectancy	Population	Continent
0	Lesotho	2598	47.1	2174645	Africa
1	Central African Republic	599	49.6	4546100	Africa
2	Swaziland	6095	51.8	1319011	Africa
3	Afghanistan	1925	53.8	33736494	Asia
4	Somalia	624	54.2	13908129	Africa

Feature extraction

```
df = pd.read_csv('titanic.csv')
age_data = df[['Age','Survived']].dropna()
kmeans = KMeans(n_clusters=4, random_state=0).fit(age_data)
age_group = kmeans.labels_
age_data['age bin'] = age_group
```





	Age	Survived	age bin
0	22.0	0	3
1	38.0	1	0
2	26.0	1	3
3	35.0	1	0
4	35.0	0	0