Ciência de Dados
(Big Data Processing and Analytics)

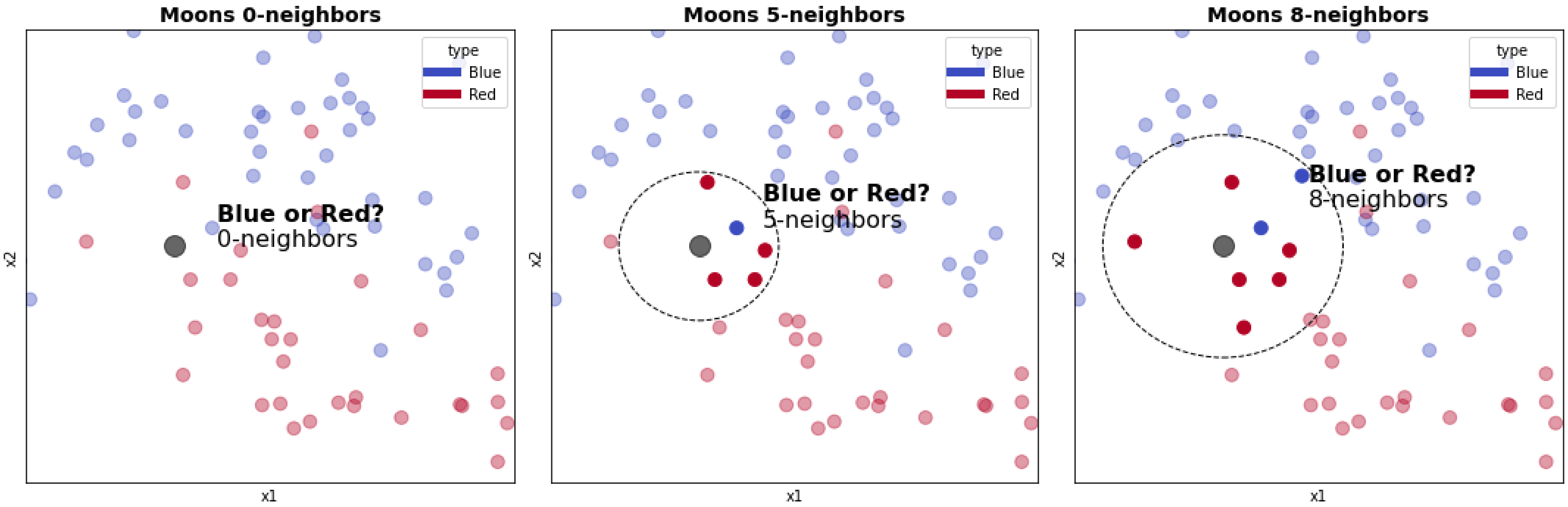
Big Data Analytics – Mineração e Análise de Dados

Professor curador
Prof. Dr. Rogério de Oliveira

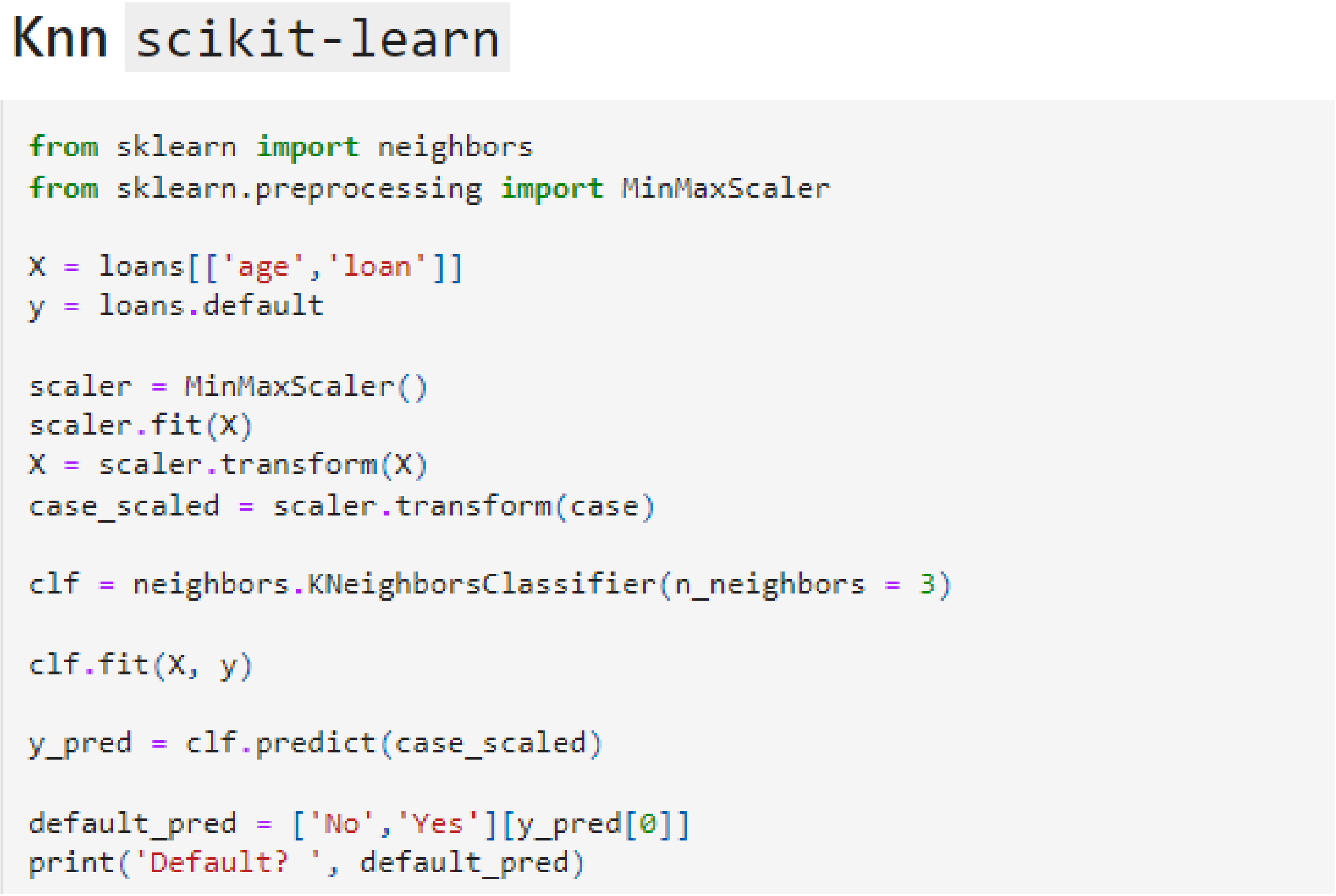
TRILHA 4
K-Vizinhos Mais Próximos, Validação Cruzada e *GridSearch*

Parte A

K Vizinhos Mais Próximos



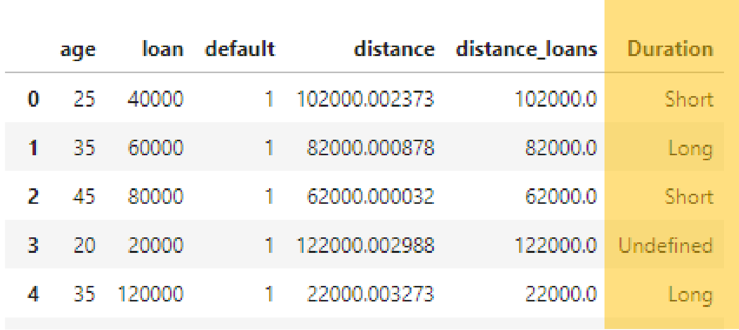
Knn com scikit-learn



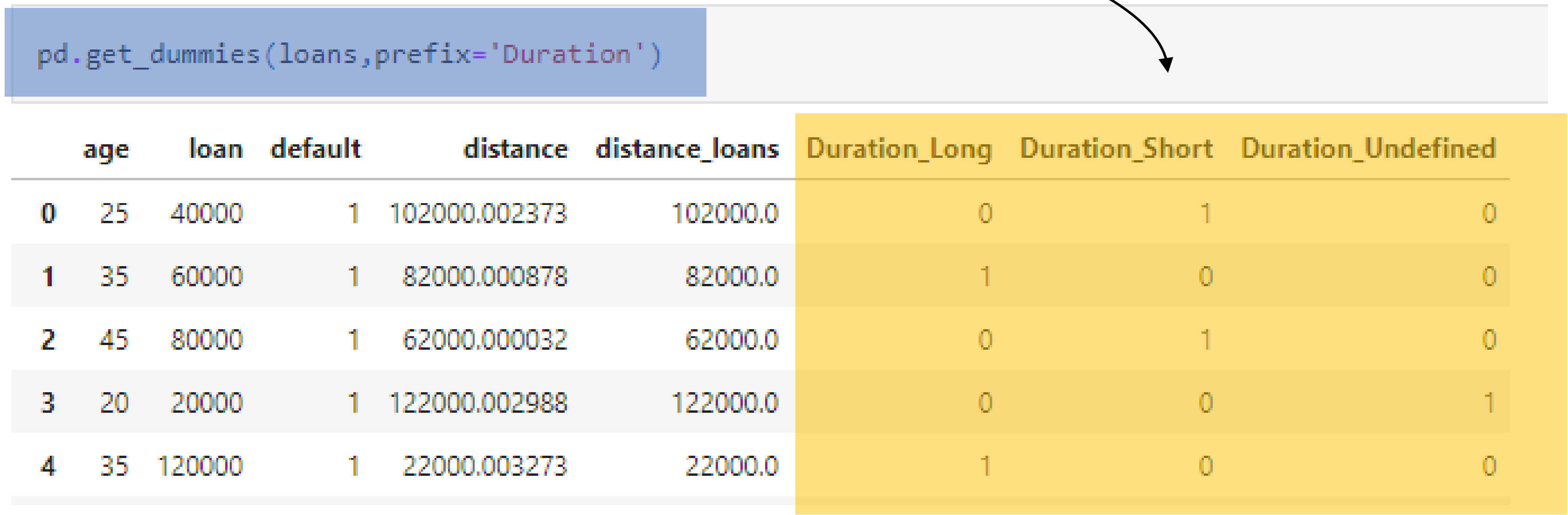
Knn `scikit-learn`

```
from sklearn import neighbors
from sklearn.preprocessing import MinMaxScaler

X = loans[['age','loan']]
y = loans.default

scaler = MinMaxScaler()
scaler.fit(X)
X = scaler.transform(X)
case_scaled = scaler.transform(case)

clf = neighbors.KNeighborsClassifier(n_neighbors = 3)

clf.fit(X, y)

y_pred = clf.predict(case_scaled)

default_pred = ['No','Yes'][y_pred[0]]
print('Default? ', default_pred)
```

Preparação dos Dados: Hot Encode



	age	loan	default	distance	distance_loans	Duration
0	25	40000	1	102000.002373	102000.0	Short
1	35	60000	1	82000.000878	82000.0	Long
2	45	80000	1	62000.000032	62000.0	Short
3	20	20000	1	122000.002988	122000.0	Undefined
4	35	120000	1	22000.003273	22000.0	Long

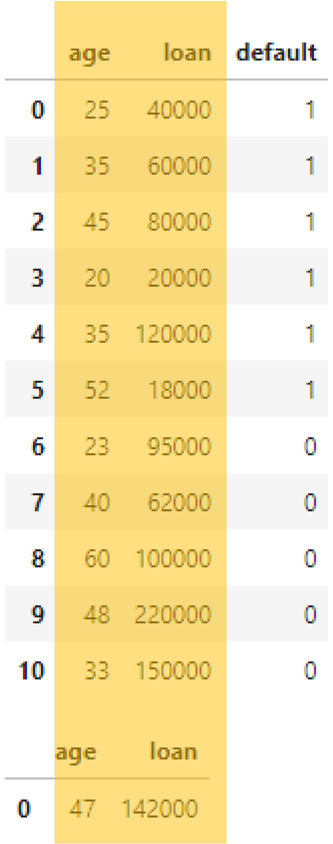


```
pd.get_dummies(loans,prefix='Duration')
```

	age	loan	default	distance	distance_loans	Duration_Long	Duration_Short	Duration_Undefined
0	25	40000	1	102000.002373	102000.0	0	1	0
1	35	60000	1	82000.000878	82000.0	1	0	0
2	45	80000	1	62000.000032	62000.0	0	1	0
3	20	20000	1	122000.002988	122000.0	0	0	1
4	35	120000	1	22000.003273	22000.0	1	0	0

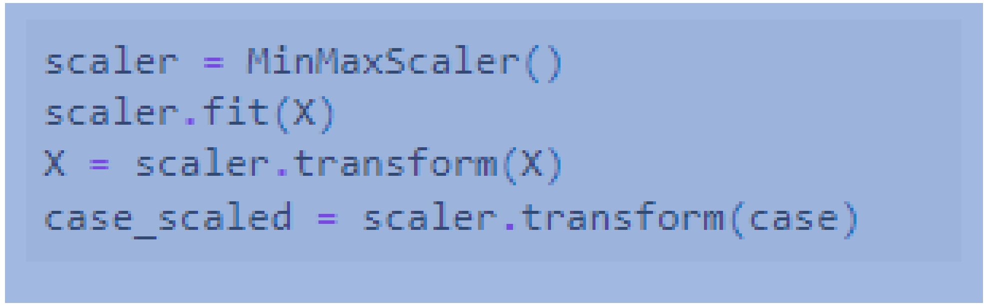
*prefira empregar o estimador do scikit-learn

Preparação dos Dados: Hot Encode

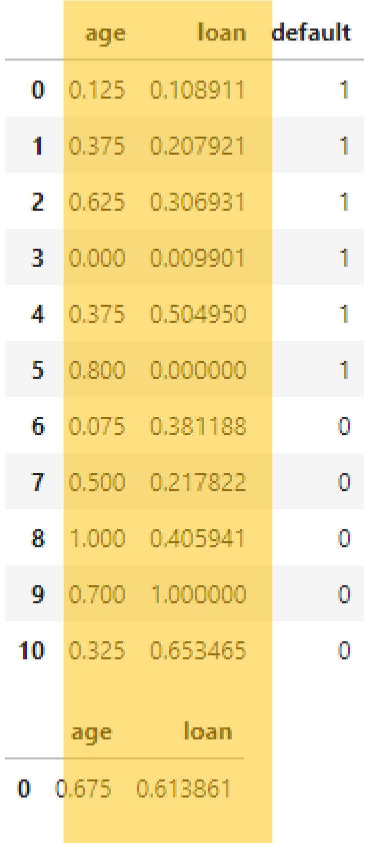


	age	loan	default
0	25	40000	1
1	35	60000	1
2	45	80000	1
3	20	20000	1
4	35	120000	1
5	52	18000	1
6	23	95000	0
7	40	62000	0
8	60	100000	0
9	48	220000	0
10	33	150000	0

	age	loan
0	47	142000



```
scaler = MinMaxScaler()
scaler.fit(X)
X = scaler.transform(X)
case_scaled = scaler.transform(case)
```



	age	loan	default
0	0.125	0.108911	1
1	0.375	0.207921	1
2	0.625	0.306931	1
3	0.000	0.009901	1
4	0.375	0.504950	1
5	0.800	0.000000	1
6	0.075	0.381188	0
7	0.500	0.217822	0
8	1.000	0.405941	0
9	0.700	1.000000	0
10	0.325	0.653465	0

	age	loan
0	0.675	0.613861

TRILHA 4
K-Vizinhos Mais Próximos, Validação Cruzada e *GridSearch*

Parte B

Métricas de Distância

$i.\, d(x,y) \geq 0$

$ii.\, d(x,x) = 0$

$iii.\, d(x,y) = d(y,x)$

$iv.\, d(x,y) \leq d(x,z) + d(z,y)$

Distância Euclidiana	$\|\|ab\|\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
Distância Euclidiana Quadrática	$\|\|ab\|\|_2^2 = \sum_i (a_i - b_i)^2$
Distância de Manhattan	$\|\|ab\|\|_1 = \sum_i \|a_i - b_i\|$
Distância Máxima	$\|\|ab\|\|_\infty = \max_i \|a_i - b_i\|$
Distância Minkowski	$\|\|ab\|\|_{Minkowski} = (\sum_i \|a_i - b_i\|^p)^{\frac{1}{p}}$

Distância Cosseno

$$i.\, d(x,y) \geq 0$$

$$ii.\, d(x,x) = 0$$

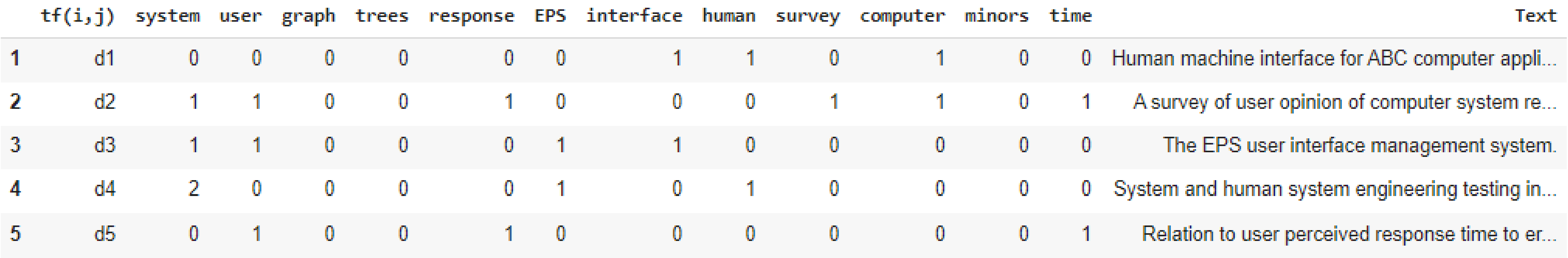
$$iii.\, d(x,y) = d(y,x)$$

$$iv.\, d(x,y) \leq d(x,z) + d(z,y)$$

$$ab^t = ||a||||b||cos(\theta)$$

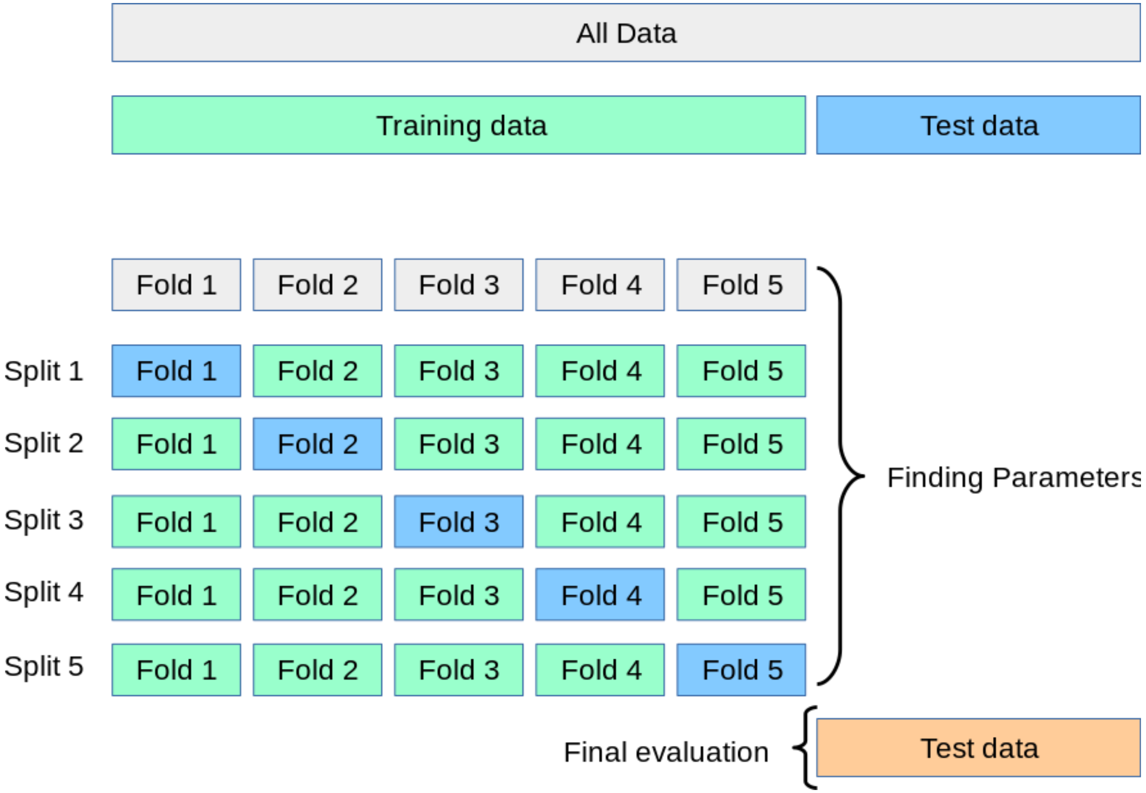
$$similarity(a,b) = cos(\theta) = \frac{ab^t}{||a||||b||} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

$$distance(a,b) = 1 - similarity(a,b)$$



	tf(i,j)	system	user	graph	trees	response	EPS	interface	human	survey	computer	minors	time	Text
1	d1	0	0	0	0	0	0	1	1	0	1	0	0	Human machine interface for ABC computer appli...
2	d2	1	1	0	0	1	0	0	0	1	1	0	1	A survey of user opinion of computer system re...
3	d3	1	1	0	0	0	1	1	0	0	0	0	0	The EPS user interface management system.
4	d4	2	0	0	0	0	1	0	1	0	0	0	0	System and human system engineering testing in...
5	d5	0	1	0	0	1	0	0	0	0	0	0	1	Relation to user perceived response time to er...

Seleção de Hiperparâmetros: Cross Validation



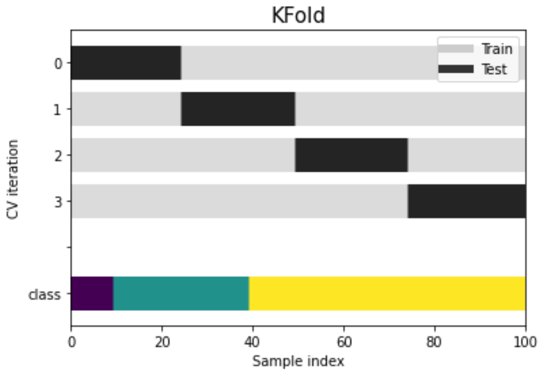
All Data

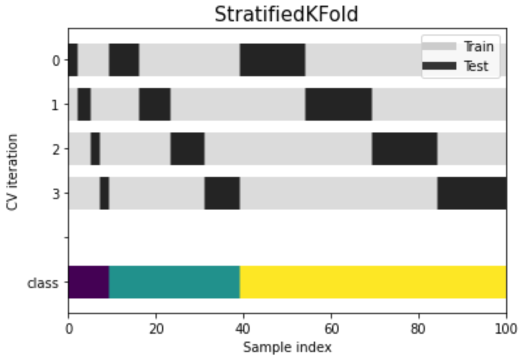
Training data | Test data

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

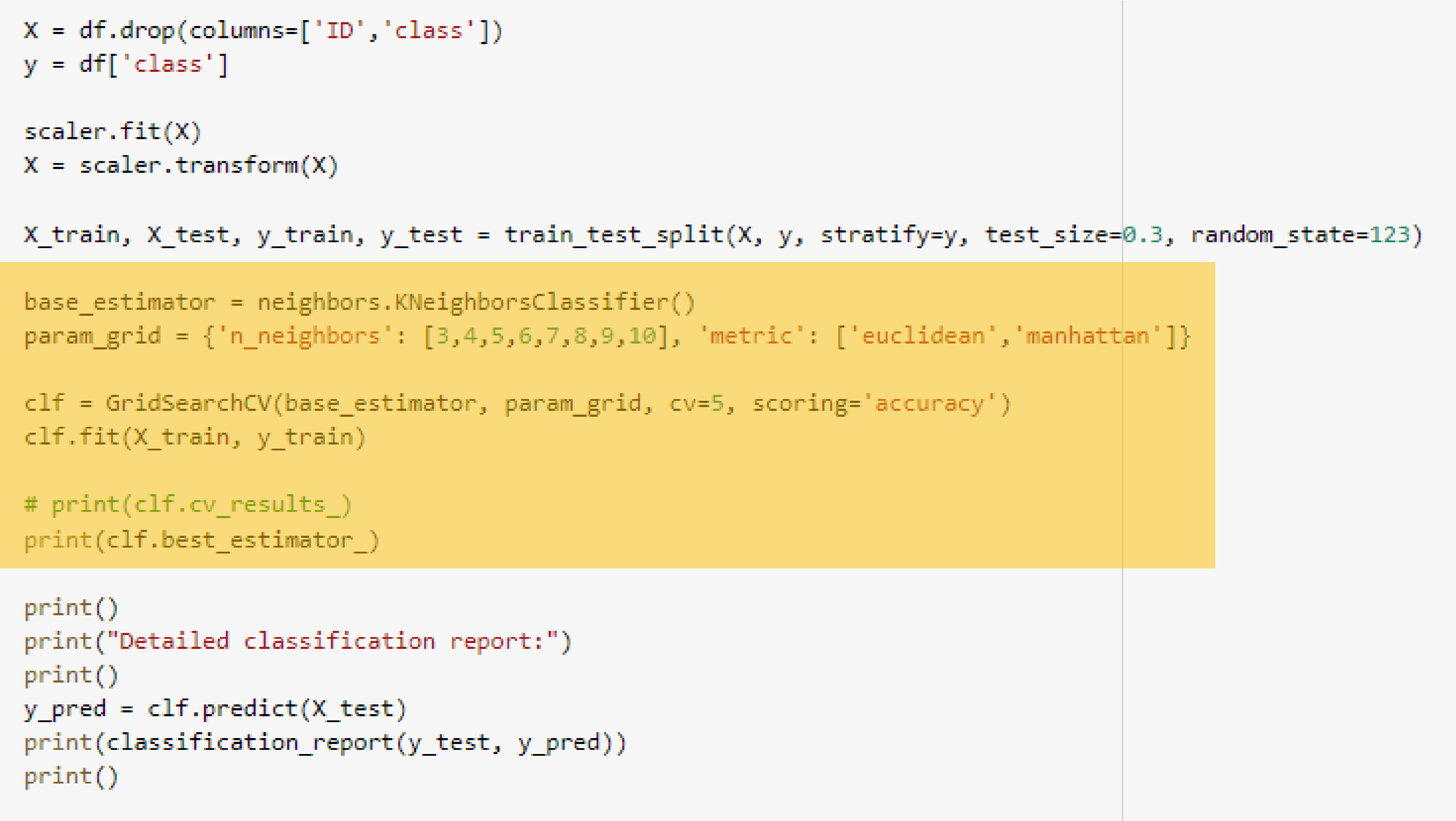
Finding Parameters

Final evaluation | Test data



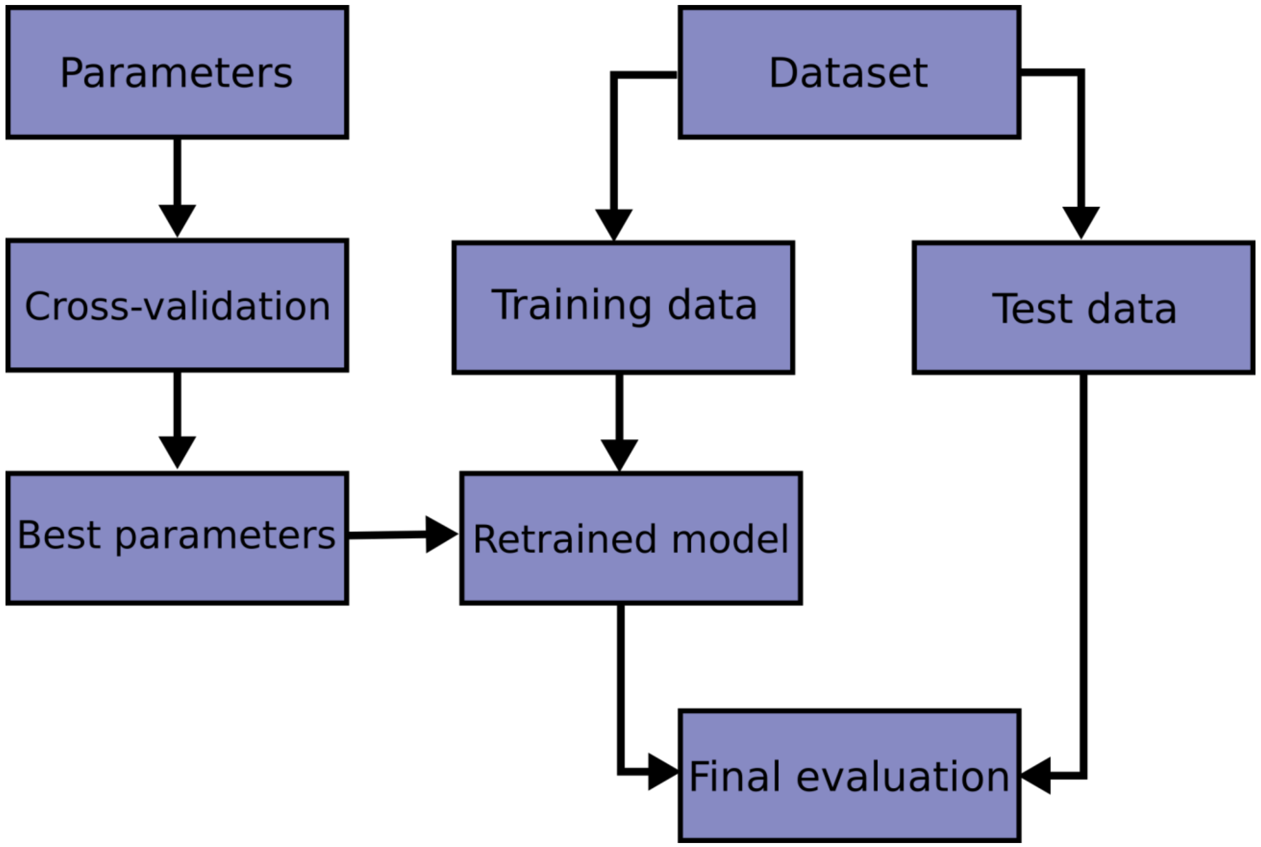


Seleção de Hiperparâmetros: GridSearch



```
X = df.drop(columns=['ID','class'])
y = df['class']

scaler.fit(X)
X = scaler.transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.3, random_state=123)

base_estimator = neighbors.KNeighborsClassifier()
param_grid = {'n_neighbors': [3,4,5,6,7,8,9,10], 'metric': ['euclidean','manhattan']}

clf = GridSearchCV(base_estimator, param_grid, cv=5, scoring='accuracy')
clf.fit(X_train, y_train)

# print(clf.cv_results_)
print(clf.best_estimator_)

print()
print("Detailed classification report:")
print()
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
print()
```

Seleção de Hiperparâmetros



EaD

Universidade Presbiteriana
Mackenzie