

# K-Nearest Neighbor

## ***K-Nearest Neighbors***

*K-Nearest neighbors for classification  
predicts the category using  
the K-closest observations*

## *K-Nearest Neighbors*

*K-Nearest neighbors for classification  
predicts the **class** using  
the K-closest observations*

### *K-Nearest Neighbors – Assumptions*

- All Predictors (features) must be numeric
- Neighbors are rows in the dataset
- There is a distance between each pair of rows
- Distance between row  $p$  and row  $q$  is

$$\sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

where  $(p_1, p_2, \dots, p_n)$  and  $(q_1, q_2, \dots, q_n)$   
are the values in the rows  $p$  and  $q$

### ***K-Nearest Neighbors – Assumptions***

$$\text{Distance} = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

- If distance is small, the rows are close to each other
- If distance is large, the rows are far from each other
- Each row has a closest neighbor
- Each row has K closest neighbors
- Each row has a category in column Y

### ***K-Nearest Neighbors – Procedure***

For each data point (row)

- Identify the K-nearest neighbors (rows)
- Count how many of them belong to each Y-category
- Identify the category with the largest count

Classify the data point as member of that category

### ***K-Nearest Neighbors – Notes***

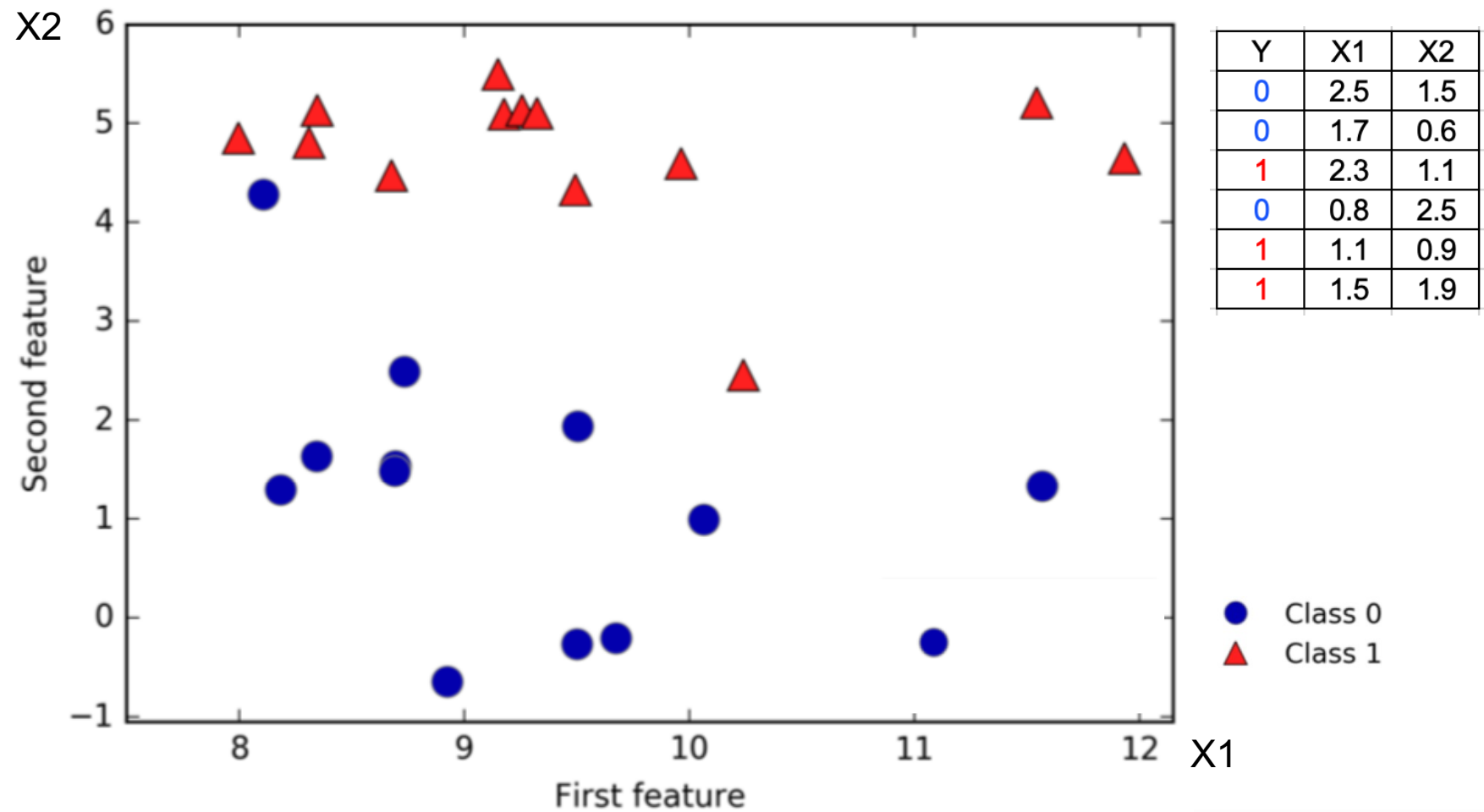
- KNN is a distance-based method
- Scaling of the data affects KNN performance
- K is a hyperparameter of the KNN Method
- Predictions depend on K
- Model performance is based on accuracy rate

***K-Nearest Neighbors - EXAMPLE***

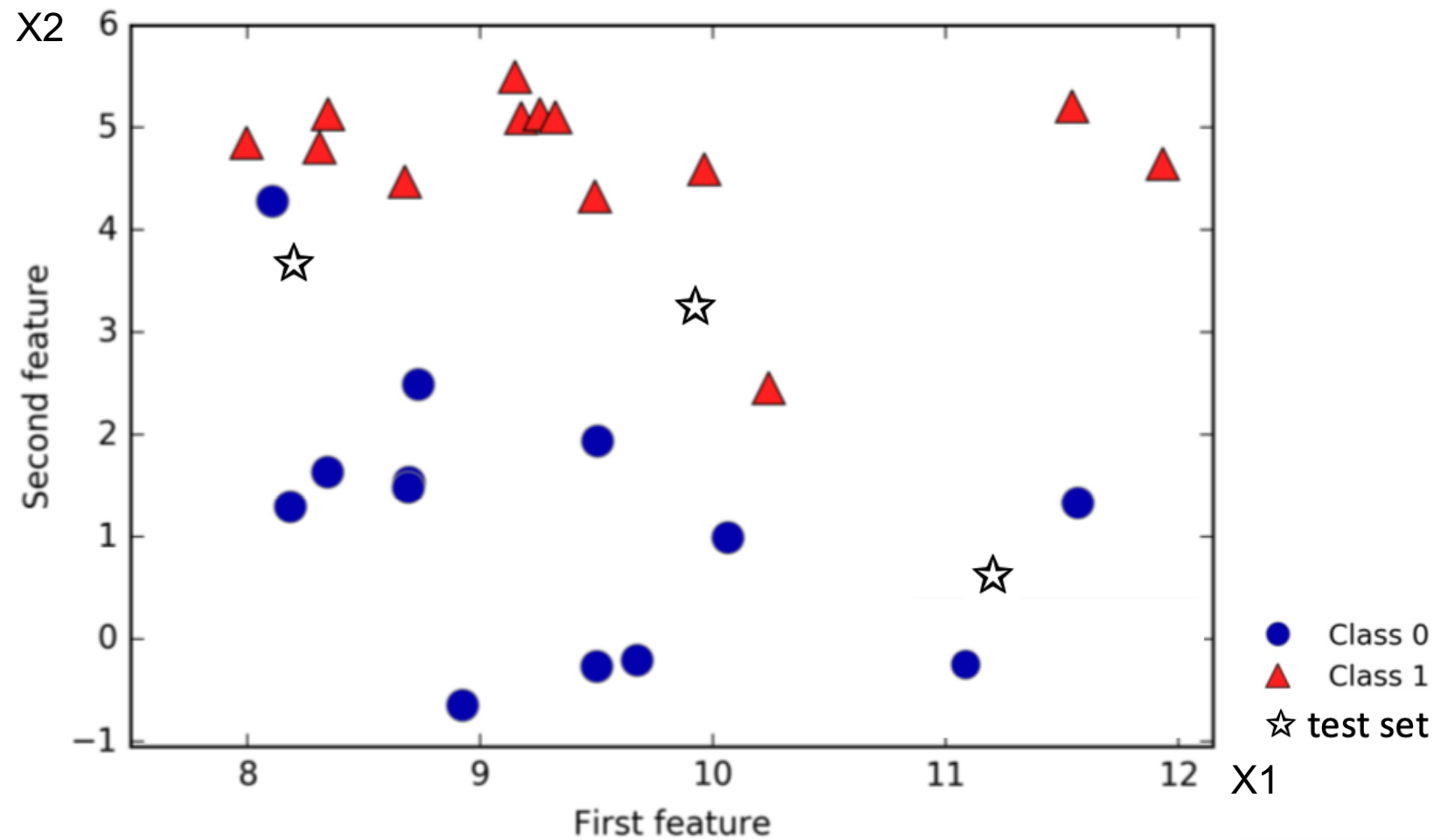
*Predict the category  
(class) of new  
observations (rows)  
using the  
3-closest neighbors*

Y	X1	X2
0	2.5	1.5
0	1.7	0.6
1	2.3	1.1
0	0.8	2.5
1	1.1	0.9
1	1.5	1.9

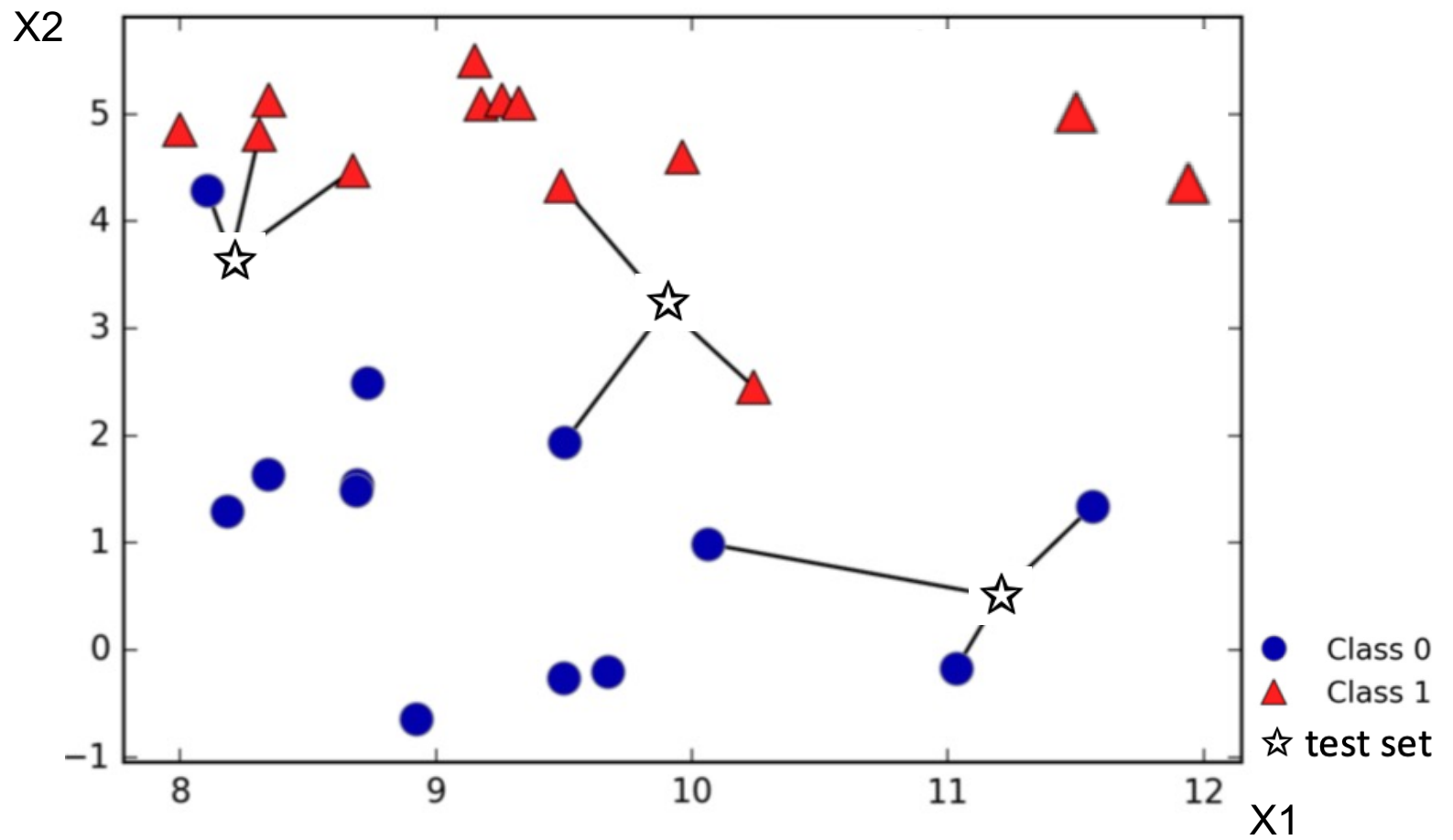


***K-Nearest Neighbors (K = 3)***

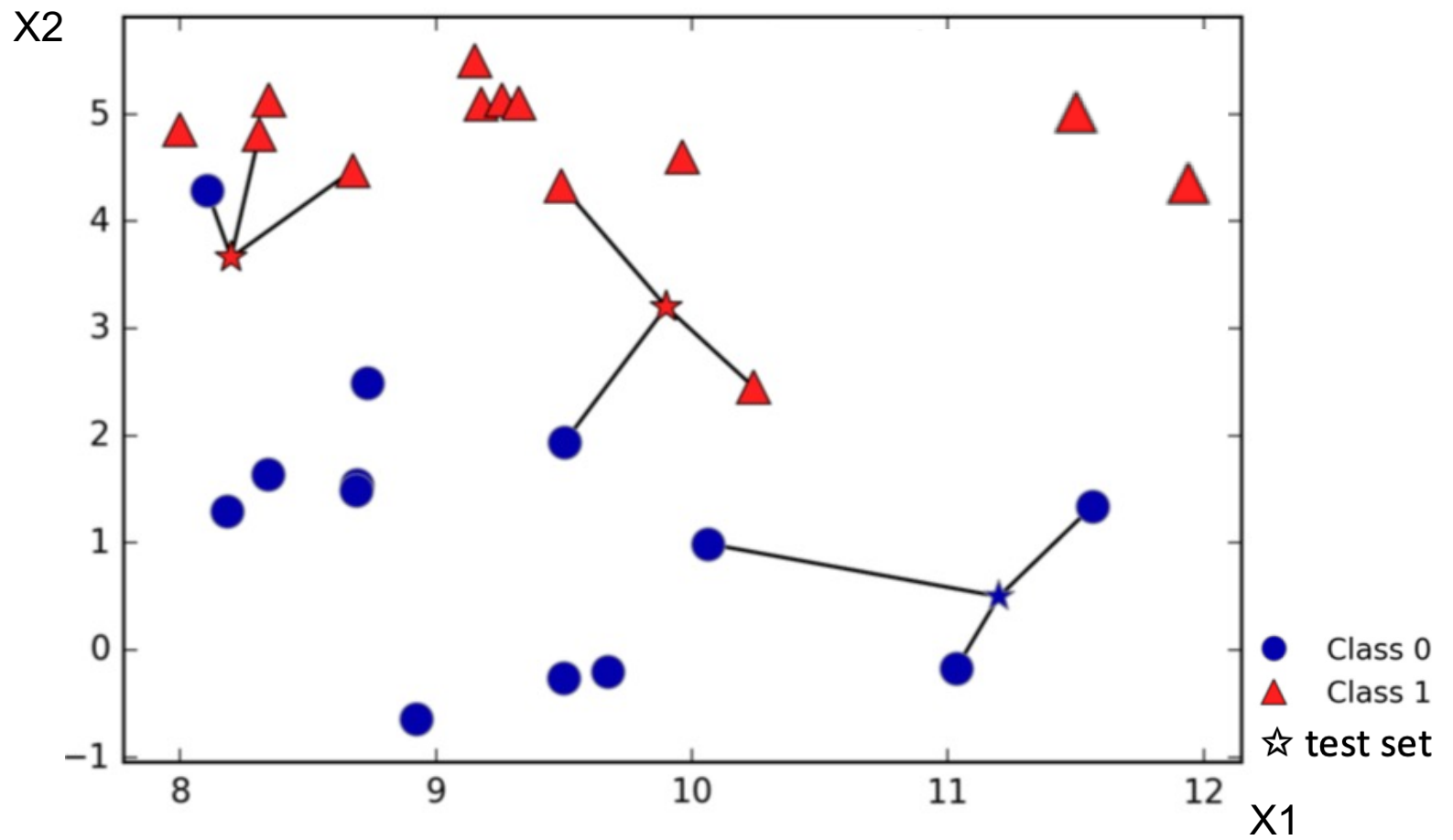
## *K-Nearest Neighbors (K = 3)*



## *K-Nearest Neighbors (K = 3)*



## *K-Nearest Neighbors (K = 3)*



***K-Nearest Neighbors - EXAMPLE***

*Predict the category  
(class) using the  
3-closest  
observations*

Y	X1	X2	Yhat
	2.5	1.5	1
	1.7	0.6	0
	2.3	1.1	1
	0.8	2.5	0
	1.1	0.9	0
	1.5	1.9	1

***K-Nearest Neighbors - EXAMPLE***

*Predict the category  
(class) using the  
3-closest  
observations*

*Compare predictions  
with the actual  
categories*

Y	X1	X2	Yhat	error?
0	2.5	1.5	1	YES
0	1.7	0.6	0	NO
1	2.3	1.1	1	NO
0	0.8	2.5	0	NO
1	1.1	0.9	0	YES
1	1.5	1.9	1	NO

Error rate = 2/6

# *Hyperparameters*

## ***PARAMETERS vs HYPERPARAMETERS***

### Parameter

Coefficient in the model that is estimated from the data

Example: Coefficients from linear regression



## ***PARAMETERS vs HYPERPARAMETERS***

### Hyperparameter

Model parameter that cannot be estimated from the data

Example: KNN (n. neighbors), RR (shrinkage parameter  $\alpha$ )

Tree (depth), RF (n. trees), GB (learning rate)

## ***PARAMETERS vs HYPERPARAMETERS***

### Hyperparameter

Model parameter that cannot be estimated from the data

Example: KNN (n. neighbors), RR (shrinkage parameter  $\alpha$ )

Tree (depth), RF (n. trees), GB (learning rate)

- A hyperparameter is a parameter whose value is used to control the learning process
- Models may have one or many hyperparameters

## ***MODELS WITH NO HYPERPARAMETERS***

If no hyperparameters, the data should be split into **2 subsets**

- Train set                      (dataset to build the model)
- Test set                        (dataset to test the model)

## ***MODELS WITH HYPERPARAMETERS***

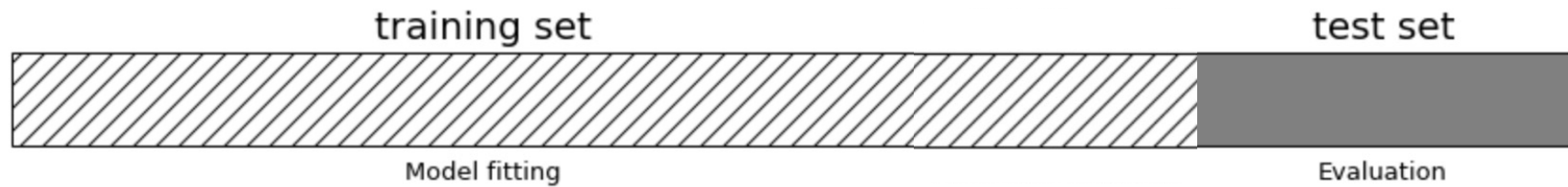
### Hyperparameter tuning

Needed to find optimal parameter values

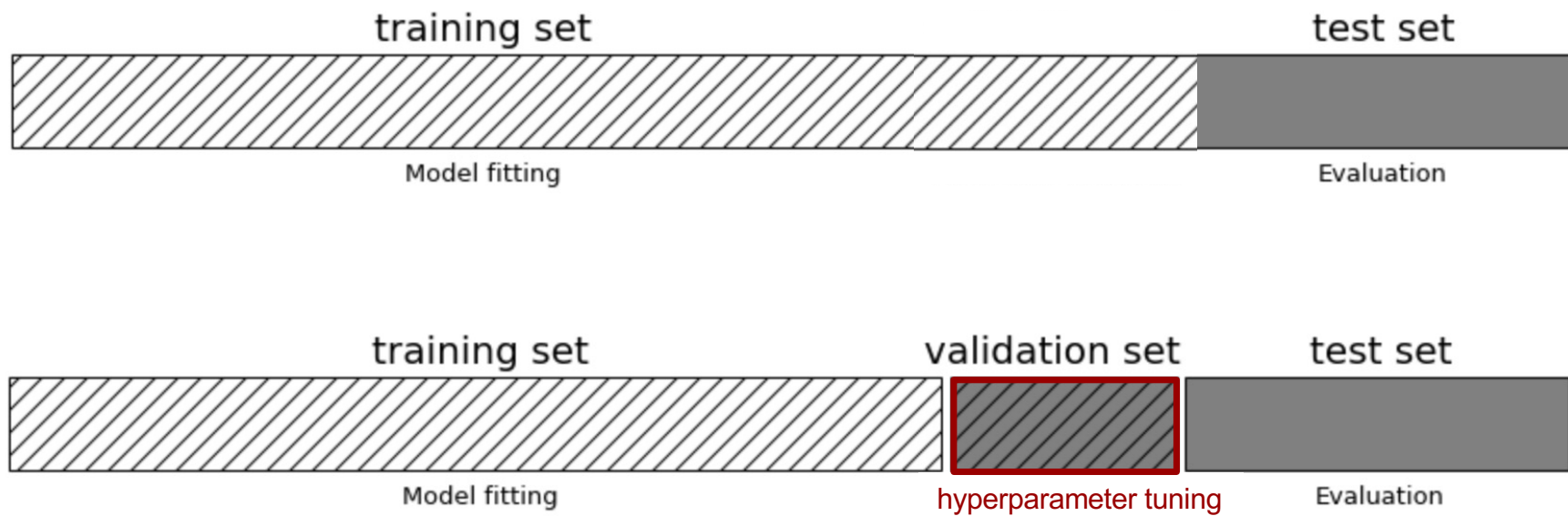
For Hyperparameter tuning the data should be split into **3 subsets**

- Train set (dataset to build the model)
- Validation set (dataset for tuning hyperparameters)
- Test set (dataset to test the model)

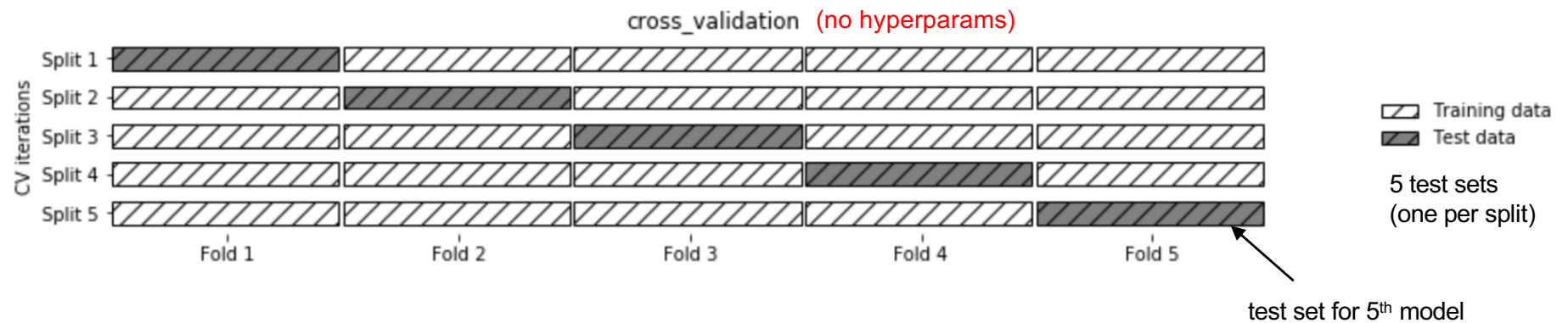
## ***HOLDOUT CROSS VALIDATION – No hyperparameters***



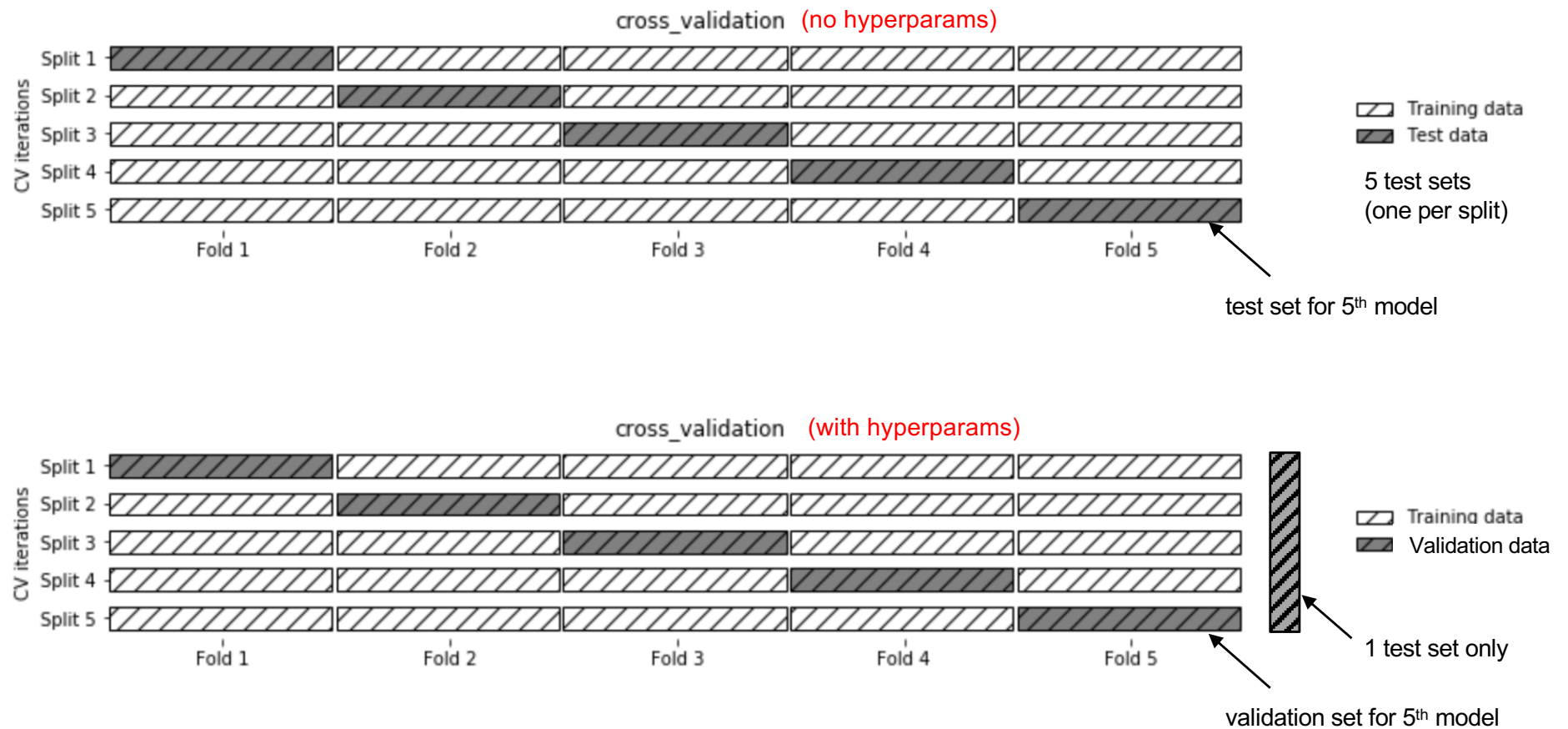
## ***HOLDOUT CROSS VALIDATION - 3 SETS***



## ***K-Fold CROSS VALIDATION – No hyperparameters***

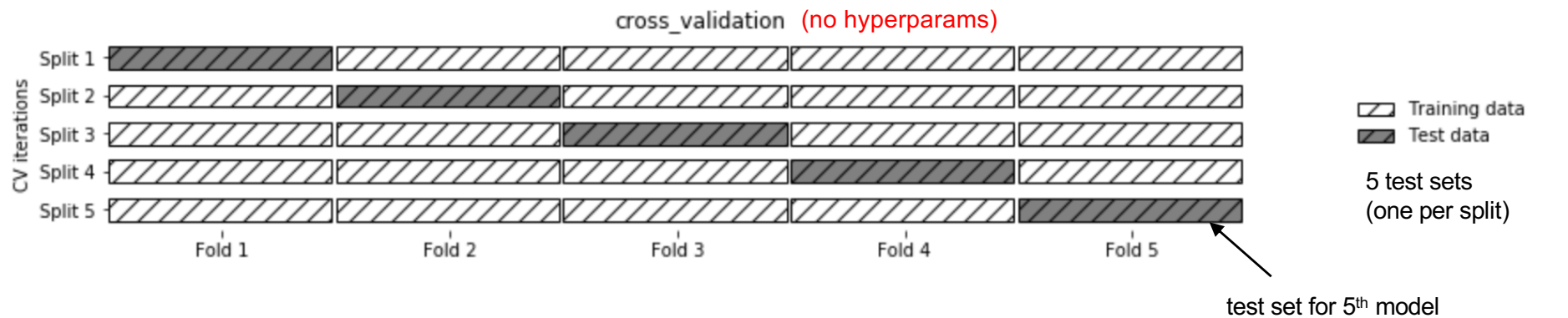


## ***K-Fold CROSS VALIDATION***

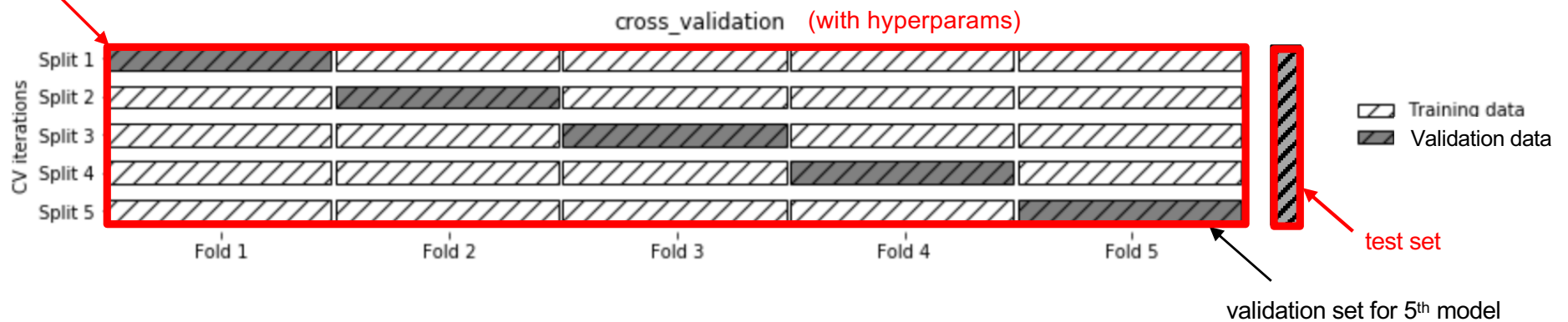




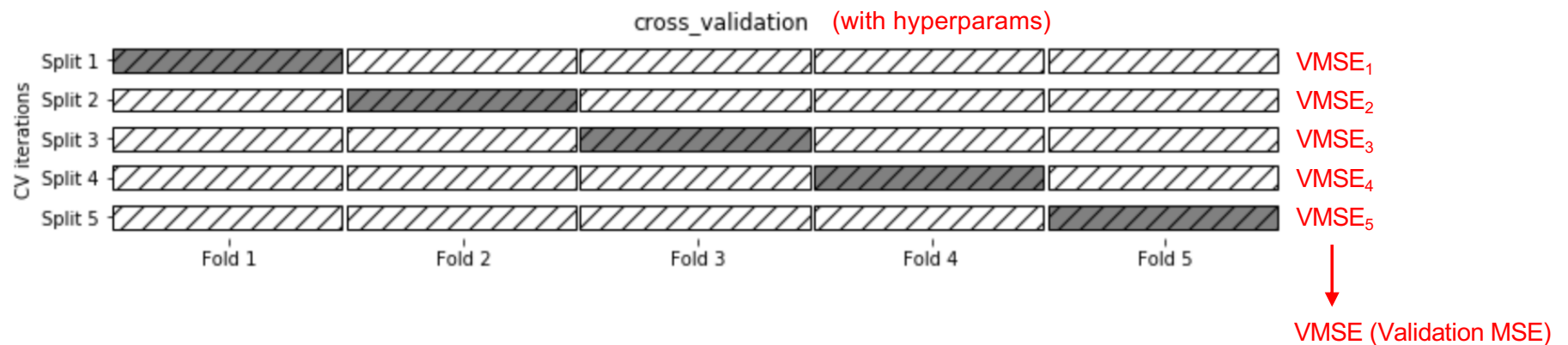
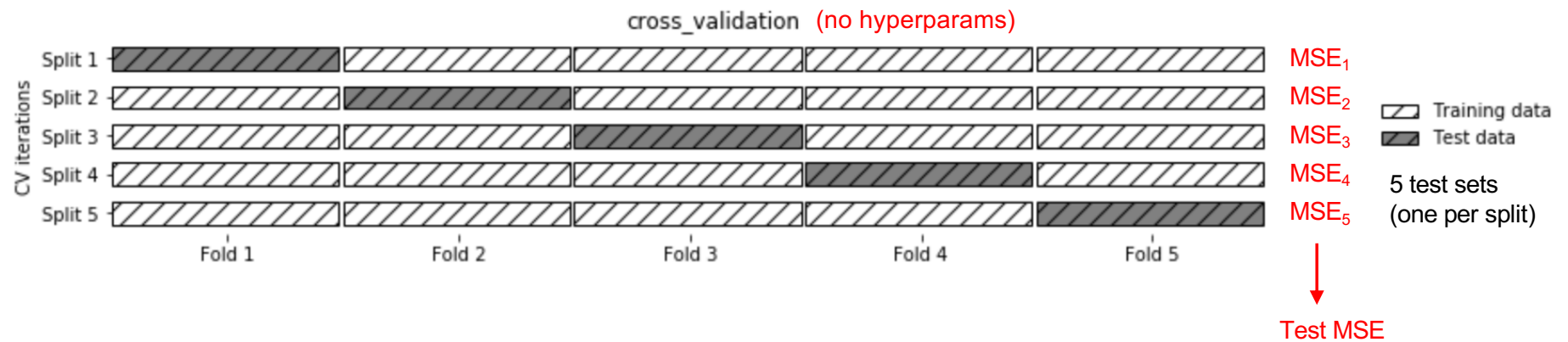
## ***K-Fold CROSS VALIDATION***



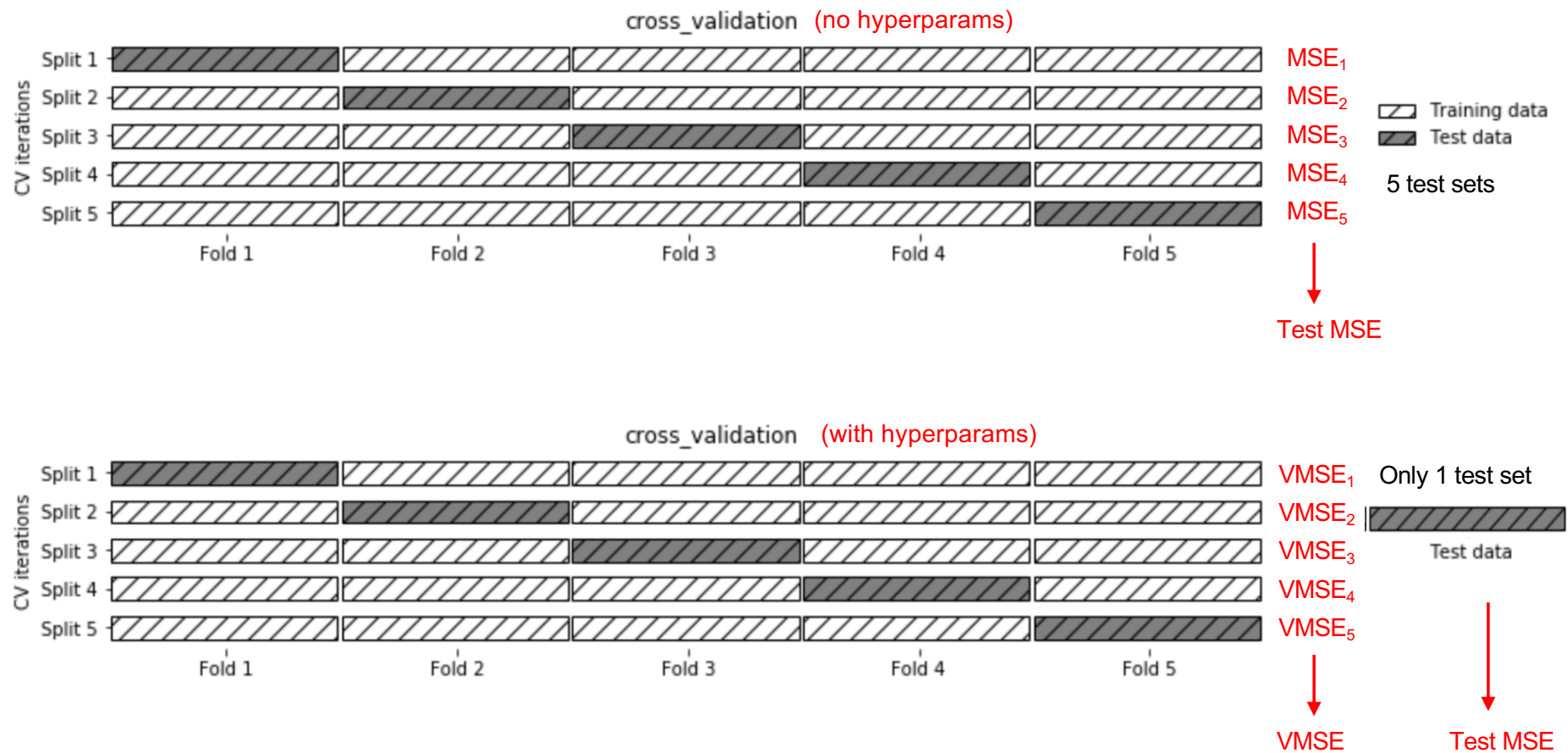
non-test set (to be split in train and validation sets)



## ***K-Fold CROSS VALIDATION***



## K-Fold CROSS VALIDATION



***K-Nearest Neighbors - EXAMPLE***

*Example – Cancer data*

### ***K-Nearest Neighbors - EXAMPLE***

- The Cancer data from *sklearn* contains lab data of 569 patients
- It includes 30 patient lab measurements associated with breast cancer tumors. **These measurements are the predictors.**
- Some patients have cancer and others have a tumor but are healthy. The target data assigns a 0 or a 1 for the patients

### ***K-Nearest Neighbors - EXAMPLE***

- The Cancer data from *sklearn* contains lab data of 569 patients
- It includes 30 patient lab measurements associated with breast cancer tumors. **These measurements are the predictors.**
- Some patients have cancer and others have a tumor but are healthy. The target variable assigns a 0 or a 1 for the patients
- Build a KNN model (with  $K=3$  nearest neighbors) to predict if the patient has cancer. Find the test accuracy rate.
- Compare Holdout Cross validation and K-fold Cross validation for **hyperparameter tuning** (the process of finding the number of neighbors  $K$  that maximizes the test accuracy rate)

## ***K-Nearest Neighbors - EXAMPLE***

Find the test accuracy rate with the following KNN models

- |   |                                     |
|---|-------------------------------------|
| 1. Holdout CV model with fixed K on original data       | ← train_test_split                  |
| 2. Holdout CV loop to find best K on original data      | } ← train_test_split<br>(two times) |
| 3. Holdout CV loop to find best K on <b>scaled data</b> |                                     |
| 4. k-Fold CV model with fixed K on original data        | } cross_val_score                   |
| 5. k-Fold CV loop to find best K on original data       |                                     |
| 6. k-Fold CV model with fixed K on <b>scaled data</b>   | } cross_val_score,<br>pipeline      |
| 7. k-Fold CV loop to find best K on <b>scaled data</b>  |                                     |

## K-Nearest Neighbors – CANCER DATASET

Y	X																			
	radius	texture	perimeter	area	average	values	concavity	concave p	symmetry	fractal_dir	radius	texture	perimeter	area	worst	values	concavity	concave p	symmetry	
M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871	25.38	17.33	184.6	2019	0.1622	0.6656	0.7119	0.2654	0.4601	
M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	24.99	23.41	158.8	1956	0.1238	0.1866	0.2416	0.186	0.275	
M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999	23.57	25.53	152.5	1709	0.1444	0.4245	0.4504	0.243	0.3613	
M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744	14.91	26.5	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	
M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	0.1809	0.05883	22.54	16.67	152.2	1575	0.1374	0.205	0.4	0.1625	0.2364	
M	12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	0.2087	0.07613	15.47	23.75	103.4	741.6	0.1791	0.5249	0.5355	0.1741	0.3985	
M	18.25	19.98	119.6	1040	0.09463	0.109	0.1127	0.074	0.1794	0.05742	22.88	27.66	153.2	1606	0.1442	0.2576	0.3784	0.1932	0.3063	
M	13.71	20.83	90.2	577.9	0.1189	0.1645	0.09366	0.05985	0.2196	0.07451	17.06	28.14	110.6	897	0.1654	0.3682	0.2678	0.1556	0.3196	
M	13	21.82	87.5	519.8	0.1273	0.1932	0.1859	0.09353	0.235	0.07389	15.49	30.73	106.2	739.3	0.1703	0.5401	0.539	0.206	0.4378	
M	12.46	24.04	83.97	475.9	0.1186	0.2396	0.2273	0.08543	0.203	0.08243	15.09	40.68	97.65	711.4	0.1853	1.058	1.105	0.221	0.4366	
M	16.02	23.24	102.7	797.8	0.08206	0.06669	0.03299	0.03323	0.1528	0.05697	19.19	33.88	123.8	1150	0.1181	0.1551	0.1459	0.09975	0.2948	
M	15.78	17.89	103.6	781	0.0971	0.1292	0.09954	0.06606	0.1842	0.06082	20.42	27.28	136.5	1299	0.1396	0.5609	0.3965	0.181	0.3792	
M	19.17	24.8	132.4	1123	0.0974	0.2458	0.2065	0.1118	0.2397	0.078	20.96	29.94	151.7	1332	0.1037	0.3903	0.3639	0.1767	0.3176	
M	15.85	23.95	103.7	782.7	0.08401	0.1002	0.09938	0.05364	0.1847	0.05338	16.84	27.66	112	876.5	0.1131	0.1924	0.2322	0.1119	0.2809	
M	13.73	22.61	93.6	578.3	0.1131	0.2293	0.2128	0.08025	0.2069	0.07682	15.03	32.01	108.8	697.7	0.1651	0.7725	0.6943	0.2208	0.3596	
M	14.54	27.54	96.73	658.8	0.1139	0.1595	0.1639	0.07364	0.2303	0.07077	17.46	37.13	124.1	943.2	0.1678	0.6577	0.7026	0.1712	0.4218	
M	14.68	20.13	94.74	684.5	0.09867	0.072	0.07395	0.05259	0.1586	0.05922	19.07	30.88	123.4	1138	0.1464	0.1871	0.2914	0.1609	0.3029	
M	16.13	20.68	108.1	798.8	0.117	0.2022	0.1722	0.1028	0.2164	0.07356	20.96	31.48	136.8	1315	0.1789	0.4233	0.4784	0.2073	0.3706	
M	19.81	22.15	130	1260	0.09831	0.1027	0.1479	0.09498	0.1582	0.05395	27.32	30.88	186.8	2398	0.1512	0.315	0.5372	0.2388	0.2768	
B	13.54	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.04781	0.1885	0.05766	15.11	19.26	99.7	711.2	0.144	0.1773	0.239	0.1288	0.2977	
B	13.08	15.71	85.63	520	0.1075	0.127	0.04568	0.0311	0.1967	0.06811	14.5	20.49	96.09	630.5	0.1312	0.2776	0.189	0.07283	0.3184	
B	9.504	12.44	60.34	273.9	0.1024	0.06492	0.02956	0.02076	0.1815	0.06905	10.23	15.66	65.13	314.9	0.1324	0.1148	0.08867	0.06227	0.245	
M	15.34	14.26	102.5	704.4	0.1073	0.2135	0.2077	0.09756	0.2521	0.07032	18.07	19.08	125.1	980.9	0.139	0.5954	0.6305	0.2393	0.4667	
M	21.16	23.04	137.2	1404	0.09428	0.1022	0.1097	0.08632	0.1769	0.05278	29.17	35.59	188	2615	0.1401	0.26	0.3155	0.2009	0.2822	

(569, )

(569, 30)



## ***K-Nearest Neighbors - EXAMPLE***

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

from sklearn.datasets import load_breast_cancer
from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import StratifiedKFold
from sklearn.pipeline import Pipeline
```

## ***K-Nearest Neighbors - EXAMPLE***

```
from sklearn.datasets import load_breast_cancer
```

```
cancer = load_breast_cancer()  
cancer.keys()
```

```
dict_keys(['data', 'target', 'frame', 'target_names',  
          'DESCR', 'feature_names', 'filename', 'data_module'])
```

## ***K-Nearest Neighbors - EXAMPLE***

```
from sklearn.datasets import load_breast_cancer
```

```
cancer = load_breast_cancer()  
cancer.keys()
```

```
dict_keys(['data', 'target', 'frame', 'target_names',  
          'DESCR', 'feature_names', 'filename', 'data_module'])
```

```
y = cancer.target  
X = cancer.data
```

***K-Nearest Neighbors - EXAMPLE***

*1. Hold out Cross validation  
-Fixed K, No Scaling-*

## *K-Nearest Neighbors – holdout cross validation (hyperparameter known $K=3$ )*

```
y = cancer.target  
X = cancer.data
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,stratify=y,  
                                                random_state=66)
```

```
print('train set:', X_train.shape,y_train.shape)  
print('test  set:', X_test.shape,y_test.shape)
```

```
train set: (426, 30) (426,)  
test  set: (143, 30) (143,)
```

```
426/569
```

```
0.7486818980667839
```

```
train_size = 0.75
```

## *K-Nearest Neighbors – holdout cross validation (K=3)*

```
model = KNeighborsClassifier(n_neighbors=3).fit(X_train, y_train)
model.predict(X_test)
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1,
       1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,
       1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

```
# Find test accuracy rate
```

```
model.score(X_test, y_test)
```

```
0.9230769230769231
```

## *K-Nearest Neighbors – Test accuracy rate (K=3)*

```
# Find test accuracy rate
model.score(X_test, y_test)
```

```
0.9230769230769231
```

```
# confusion matrix
yhat = model.predict(X_test)
pd.crosstab(y_test,yhat,
            rownames=['y_test'],colnames=['predictions'])
```

	predictions	
y_test	0	1

0	47	6
---	----	---

← Six 'category 0' patients predicted as category 1

1	5	85
---	---	----

← Five 'category 1' patients predicted as category 0

```
(47+85)/143
```

```
accuracy rate
```

```
0.9230769230769231
```

***K-Nearest Neighbors - EXAMPLE***

*2. Hold out Cross validation  
-Search for K, No Scaling-*



## *K-Nearest Neighbors – Holdout CrossValidation to select number of neighbors K*

```
y0 = cancer.target  
X0 = cancer.data  
X0.shape
```

```
(569, 30)
```

Reserve test set until last step →

```
X,X_test,y,y_test = train_test_split(X0,y0,stratify=y,  
                                     random_state=66)
```

Split remaining data into  
train and validation sets →

```
X_train,X_validation,\  
y_train,y_validation = train_test_split(X,y,stratify=y,  
                                       random_state=66)
```

```
print(X_train.shape, X_validation.shape)
```

```
(319, 30) (107, 30)
```

## *K-Nearest Neighbors – Holdout CrossValidation to select number of neighbors K*

```
validation_accuracy = []  
  
for k in range(1,20):  
    model = KNeighborsClassifier(n_neighbors=k)  
    model.fit(X_train, y_train)  
    validation_accuracy.append(model.score(X_validation,  
                                           y_validation))
```

```
df = pd.DataFrame(validation_accuracy,  
                  columns = ['val_acc'])  
df.index = range(1,20)  
df[:11]
```

	val_acc
1	0.906542
2	0.915888
3	0.897196
4	0.906542
5	0.906542
6	0.915888
7	0.906542
8	0.925234
9	0.925234
10	0.925234
11	0.906542

## *K-Nearest Neighbors – Holdout CrossValidation to select number of neighbors K*

```
df.plot()  
plt.ylabel("Validation Accuracy rate")  
plt.xlabel("number of neighbors k")  
plt.legend('')  
plt.ylim(0.88,0.94)  
plt.annotate('0.925', (7.25, 0.927))
```

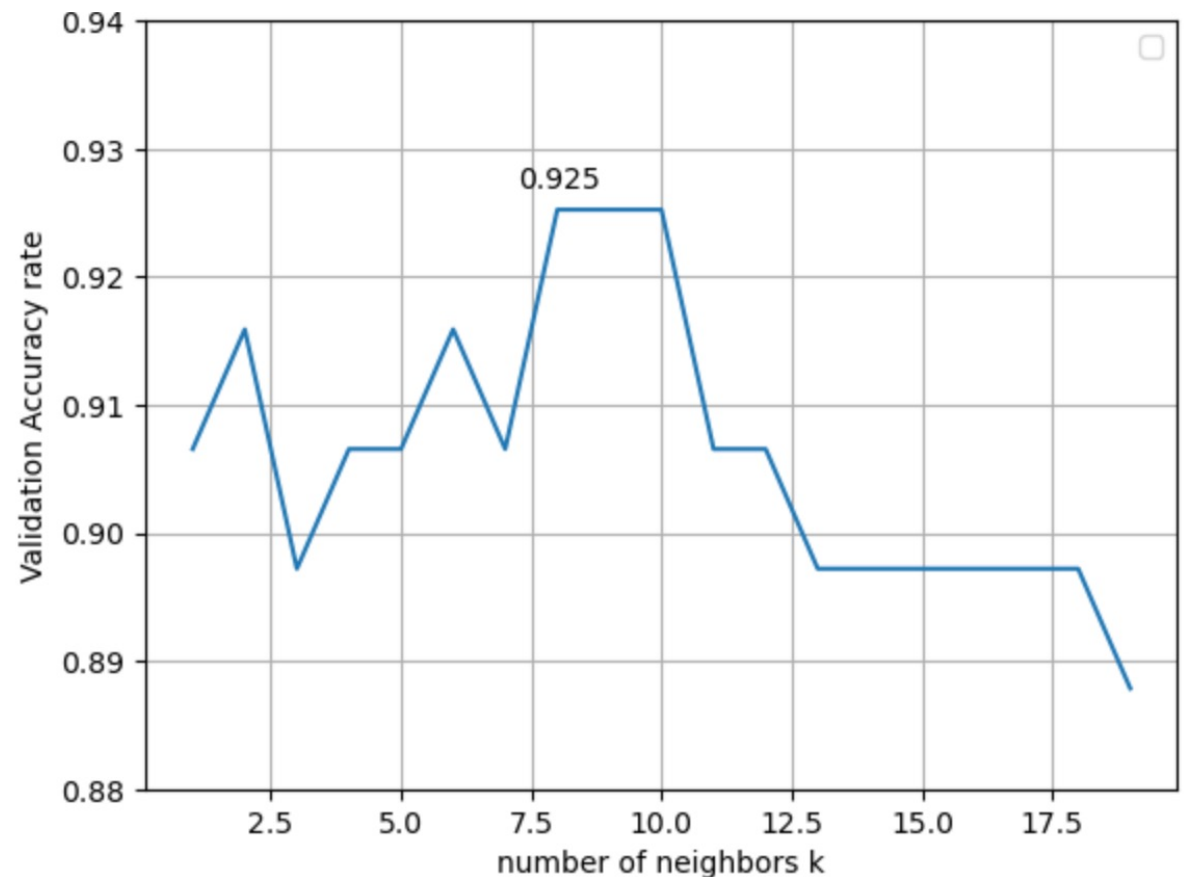
```
# test accuracy rate
```

```
# rebuild model on the  
# combined (train+validation) set
```

```
model = KNeighborsClassifier(n_neighbors=8)  
model.fit(X, y);
```

```
model.score(X_test,y_test)
```

```
0.9300699300699301
```



***K-Nearest Neighbors - EXAMPLE***

*3. Hold out Cross validation*  
*-Search for K, Scaling the data-*

## *K-Nearest Neighbors – Scaling the data*

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

# Find min/max of each feature in Train set
scaler.fit(X_train);

# then transform data into (0,1)
# by subtracting the train set Min,
# and dividing by the train set range

X_train_scaled = scaler.transform(X_train)
X_validation_scaled = scaler.transform(X_validation)
```

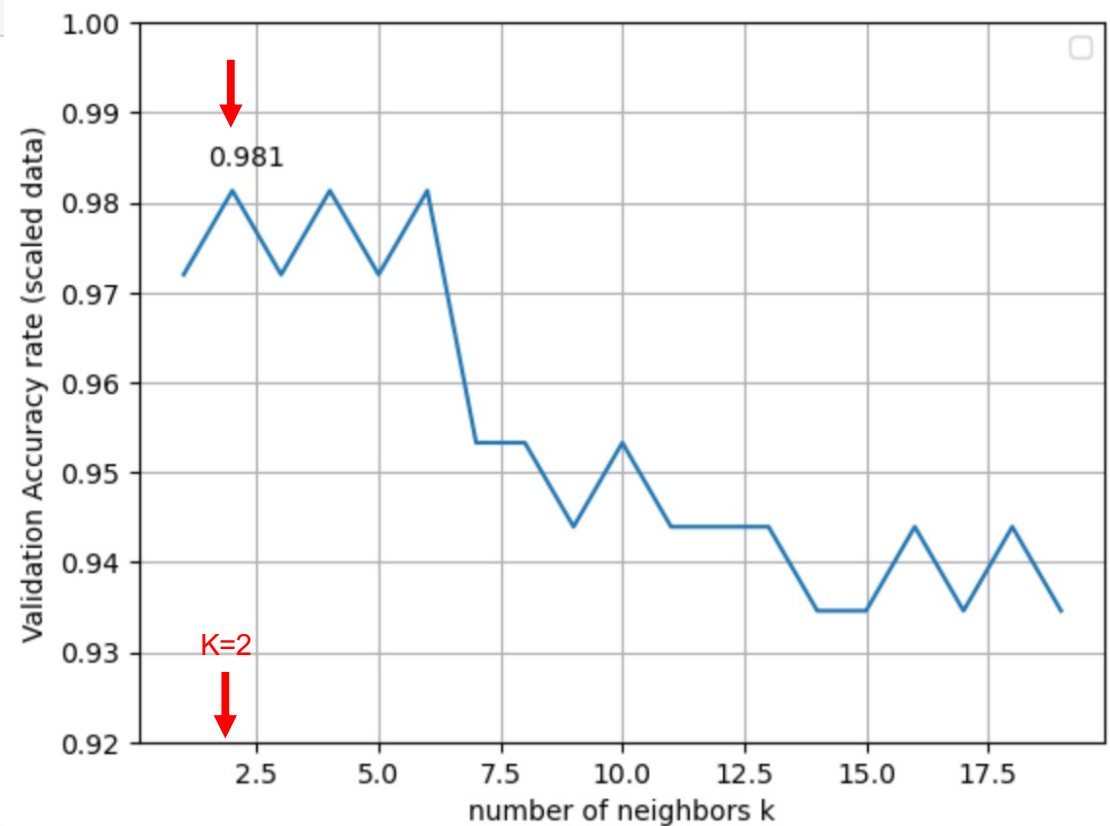
## *K-Nearest Neighbors – Holdout CrossValidation to find best K*

```
validation_accuracy_scaled = []  
  
for k in range(1,20):  
    model = KNeighborsClassifier(n_neighbors=k)  
    model.fit(X_train_scaled, y_train)  
    score = model.score(X_validation_scaled, y_validation)  
    validation_accuracy_scaled.append(score)
```

## *K-Nearest Neighbors – Holdout CrossValidation to select number of neighbors K*

```
df = pd.DataFrame(validation_accuracy_scaled,  
                  columns = ['validation_acc_scaled'])  
df.index = range(1,20)  
df[:11]
```

validation_acc_scaled	
1	0.971963
2	0.981308
3	0.971963
4	0.981308
5	0.971963
6	0.981308
7	0.953271
8	0.953271
9	0.943925
10	0.953271
11	0.943925



## ***K-Nearest Neighbors – Holdout CrossValidation to find best K***

```
# Find test accuracy rate
```

```
scaler = MinMaxScaler()  
scaler.fit(X);  
  
X_scaled = scaler.transform(X)  
X_test_scaled = scaler.transform(X_test)
```

```
# rebuild model on the  
# combined (train+validation) set  
model2 = KNeighborsClassifier(n_neighbors=2)  
model2.fit(X_scaled, y);
```

```
model2.score(X_test_scaled, y_test)
```

```
0.9230769230769231
```



***K-Nearest Neighbors - EXAMPLE***

*4. k-Fold Cross validation*  
*-Fixed K, No Scaling-*

## *K-Nearest Neighbors – 5-fold Cross Validation (no scaling) with known $K = 3$*

```
y = cancer.target  
X = cancer.data
```

```
kfold = StratifiedKFold(n_splits = 5, shuffle = True, random_state=1)
```

```
model1 = KNeighborsClassifier(n_neighbors=3)  
scores = cross_val_score(model1, X, y, cv=kfold)
```

```
# display all 5 test accuracy rates  
scores
```

```
array([0.94736842, 0.89473684, 0.92982456, 0.94736842, 0.95575221])
```

```
# average test accuracy rate  
scores.mean()
```

```
0.935010091600683
```

***K-Nearest Neighbors - EXAMPLE***

***5. k-Fold Cross validation  
-Search for K, No Scaling-***

## *K-Nearest Neighbors – 5-fold Cross Validation (no scaling) to select best K*

```
y = cancer.target  
X = cancer.data
```

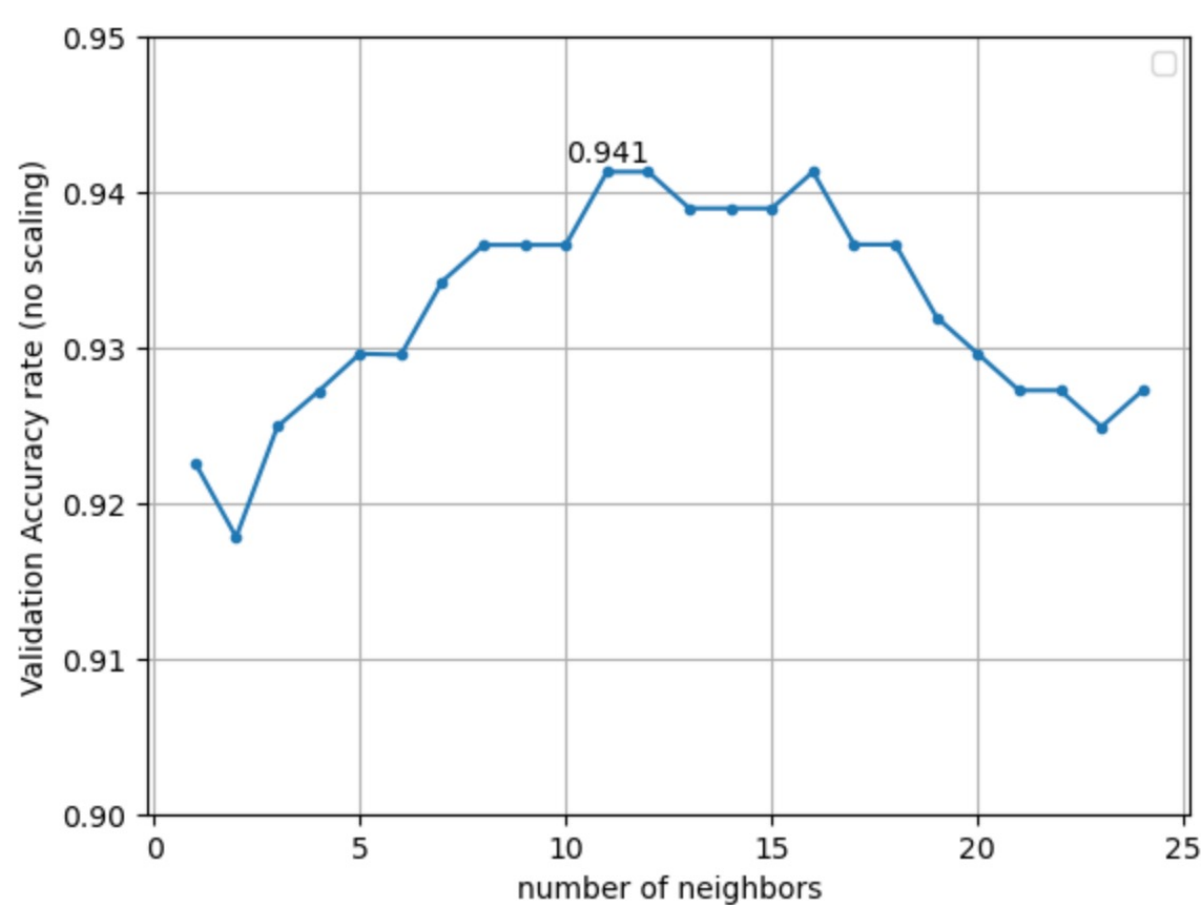
```
X_train,X_test,\  
y_train,y_test = train_test_split(X,y,  
                                  stratify=y,  
                                  random_state=66)
```

```
val_acc_rate = []  
for k in range(1,25):  
    model1 = KNeighborsClassifier(n_neighbors=k)  
    scores = cross_val_score(model1,X_train,y_train,cv=kfold)  
    val_acc_rate.append(scores.mean())
```

```
df1 = pd.DataFrame(val_acc_rate,  
                   columns=['val_acc_rate'])  
df1.index = range(1,25)
```



to be split  
into 5 folds

***K-Nearest Neighbors – 5-fold Cross Validation (no scaling) to select best K***

	average val_acc_rate
1	0.922599
2	0.917811
3	0.924925
4	0.927196
5	0.929603
6	0.929549
7	0.934254
8	0.936607
9	0.936607
10	0.936607
11	0.941313

***K-Nearest Neighbors – 5-fold Cross Validation (no scaling) test accuracy rate***

```
# Find test accuracy rate
```

```
# rebuild model on the  
# combined (train+validation) set
```

```
model2 = KNeighborsClassifier(n_neighbors=11)  
model2.fit(X_train, y_train);
```

```
model2.score(X_test, y_test)
```

```
0.916083916083916
```

average  
val\_acc\_rate

1	0.922599
2	0.917811
3	0.924925
4	0.927196
5	0.929603
6	0.929549
7	0.934254
8	0.936607
9	0.936607
10	0.936607
11	0.941313

***K-Nearest Neighbors - EXAMPLE***

*6. k-Fold Cross validation*  
*-Fixed K, Scaling the data-*

## *K-Nearest Neighbors – 5-fold Cross Validation (K = 3)*

```
y = cancer.target  
X = cancer.data
```

```
kfold = StratifiedKFold(n_splits = 5, shuffle = True, random_state=1)
```

```
scaler = MinMaxScaler()  
model1 = KNeighborsClassifier(n_neighbors=3)  
pipel = Pipeline([('transformer1', scaler), ('estimator1', model1)])
```

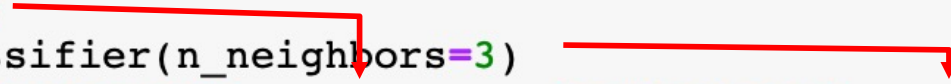


## *K-Nearest Neighbors – 5-fold Cross Validation (K = 3)*

```
y = cancer.target  
X = cancer.data
```

```
kfold = StratifiedKFold(n_splits = 5, shuffle = True, random_state=1)
```

```
scaler = MinMaxScaler()  
model1 = KNeighborsClassifier(n_neighbors=3)  
pipe1 = Pipeline([('transformer1', scaler), ('estimator1', model1)])
```



A red line originates from the `scaler` parameter in the `Pipeline` constructor, extends to the right, and then turns downward as an arrow pointing to the `estimator1` parameter, which is `model1`. This illustrates the data flow from the transformer to the estimator within the pipeline.

## *K-Nearest Neighbors – 5-fold Cross Validation (hyperparameter known $K = 3$ )*

```
y = cancer.target
X = cancer.data

kfold = StratifiedKFold(n_splits = 5, shuffle = True, random_state=1)

scaler = MinMaxScaler()
model1 = KNeighborsClassifier(n_neighbors=3)
pipe1 = Pipeline([('transformer1', scaler), ('estimator1', model1)])
scores = cross_val_score(pipe1, X, y, cv=kfold)
scores

array([0.95614035, 0.99122807, 0.98245614, 0.96491228, 0.96460177])

# test accuracy rate
scores.mean()

0.9718677224033534
```

***K-Nearest Neighbors - EXAMPLE***

*7. k-Fold Cross validation*  
*-Search for K, Scaling the data-*

## *K-Nearest Neighbors – 5-fold Cross Validation to find best K*

Reserve test set  
until last step →

```
y = cancer.target  
X = cancer.data
```

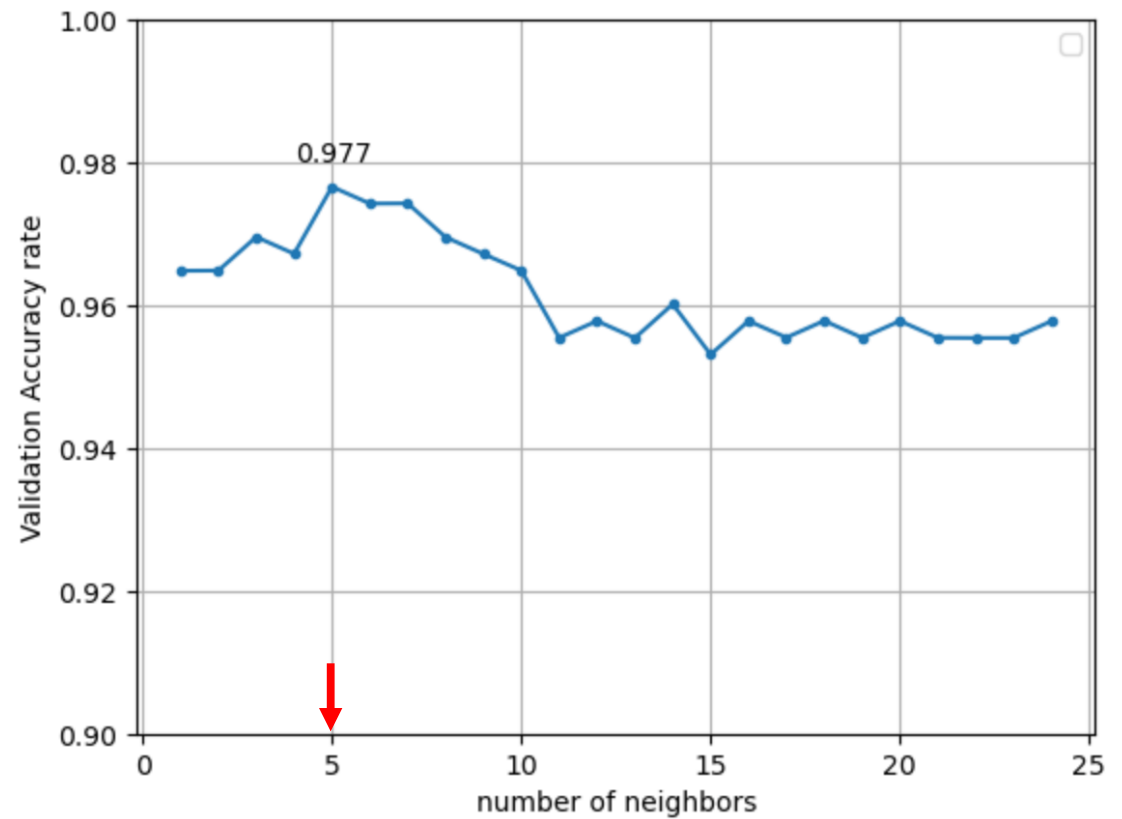
```
X_train, X_test, \  
y_train, y_test = train_test_split(X, y, stratify=y,  
                                   random_state=66)
```

```
val_acc_rate = []  
for k in range(1, 25):  
    scaler = MinMaxScaler()  
    model1 = KNeighborsClassifier(n_neighbors=k)  
    pipe1 = Pipeline([('transformer1', scaler),  
                      ('estimator1', model1)])  
    scores = cross_val_score(pipe1, X_train, y_train, cv=kfold)  
    val_acc_rate.append(scores.mean())
```

```
df1 = pd.DataFrame(val_acc_rate,  
                   columns=['val_acc_rate'])
```

## *K-Nearest Neighbors – 5-fold Cross Validation to find best K*

	val_acc_rate
1	0.964843
2	0.964870
3	0.969576
4	0.967223
5	0.976607
6	0.974254
7	0.974282
8	0.969549
9	0.967196
10	0.964870
11	0.955458



## *K-Nearest Neighbors – 5-fold Cross Validation test accuracy rate*

Reserve test set  
until last step →

```
y = cancer.target  
X = cancer.data
```

```
X_train,X_test,\  
y_train,y_test = train_test_split(X,y,stratify=y,  
                                  random_state=66)
```

## *K-Nearest Neighbors – 5-fold Cross Validation test accuracy rate*

Reserve test set  
until last step →

```
y = cancer.target  
X = cancer.data
```

```
X_train,X_test,\  
y_train,y_test = train_test_split(X,y,stratify=y,  
                                  random_state=66)
```

```
scaler = MinMaxScaler()  
scaler.fit(X_train);
```

```
X_train_scaled = scaler.transform(X_train)  
X_test_scaled = scaler.transform(X_test)
```

```
# rebuild model on the  
# combined (train+validation) set
```

```
model2 = KNeighborsClassifier(n_neighbors=5)  
model2.fit(X_train_scaled, y_train);
```

```
model2.score(X_test_scaled,y_test)
```

```
0.9440559440559441
```