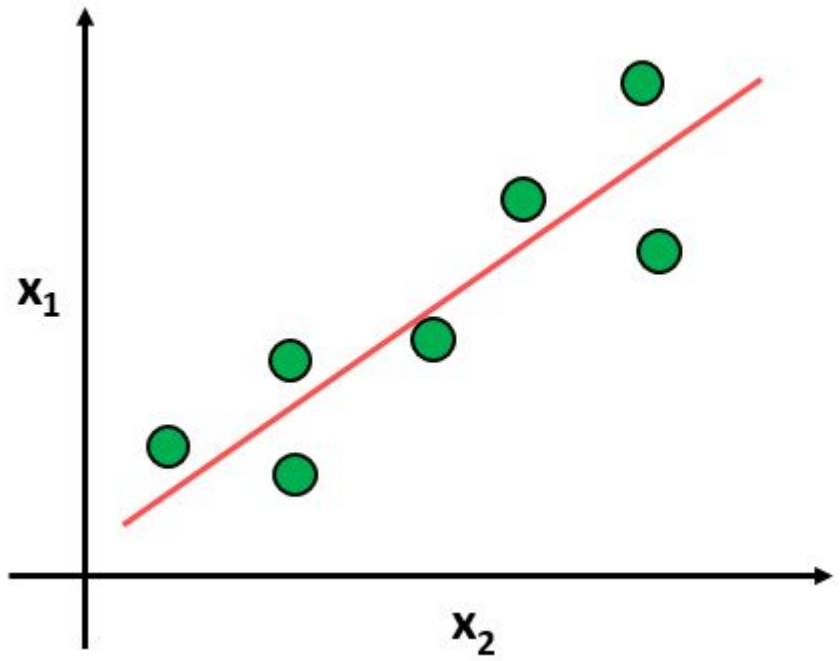
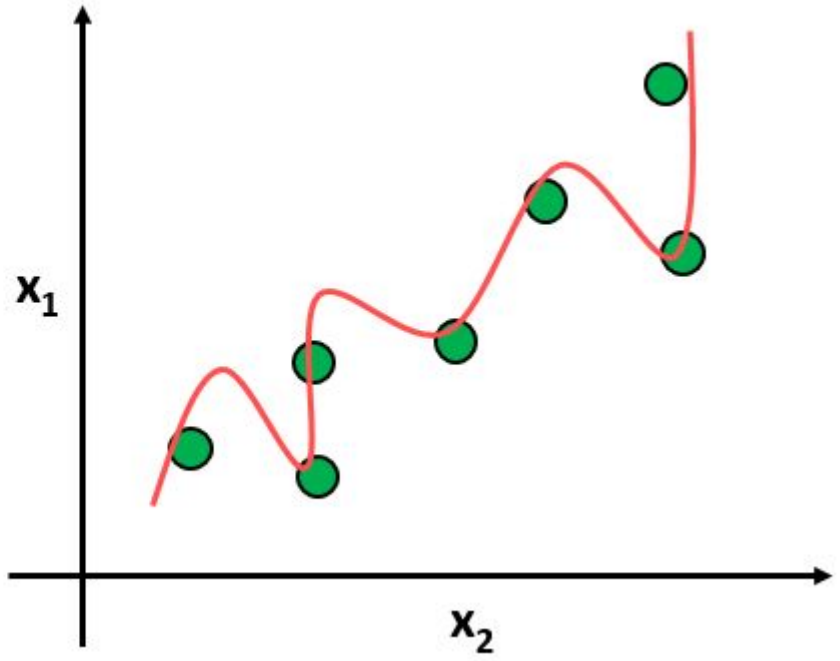
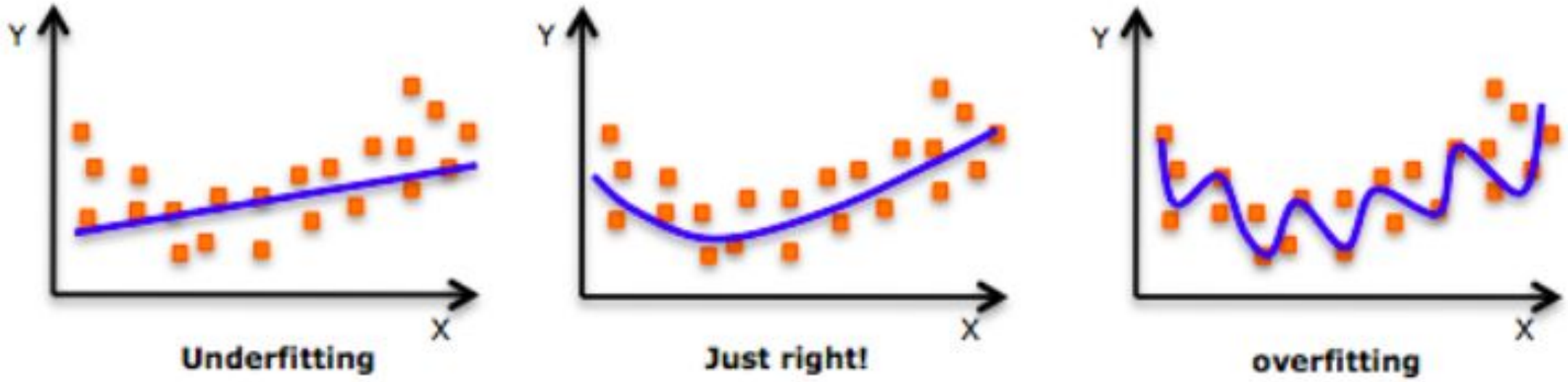
Overfit and Underfit

Model Complexity vs Generalization

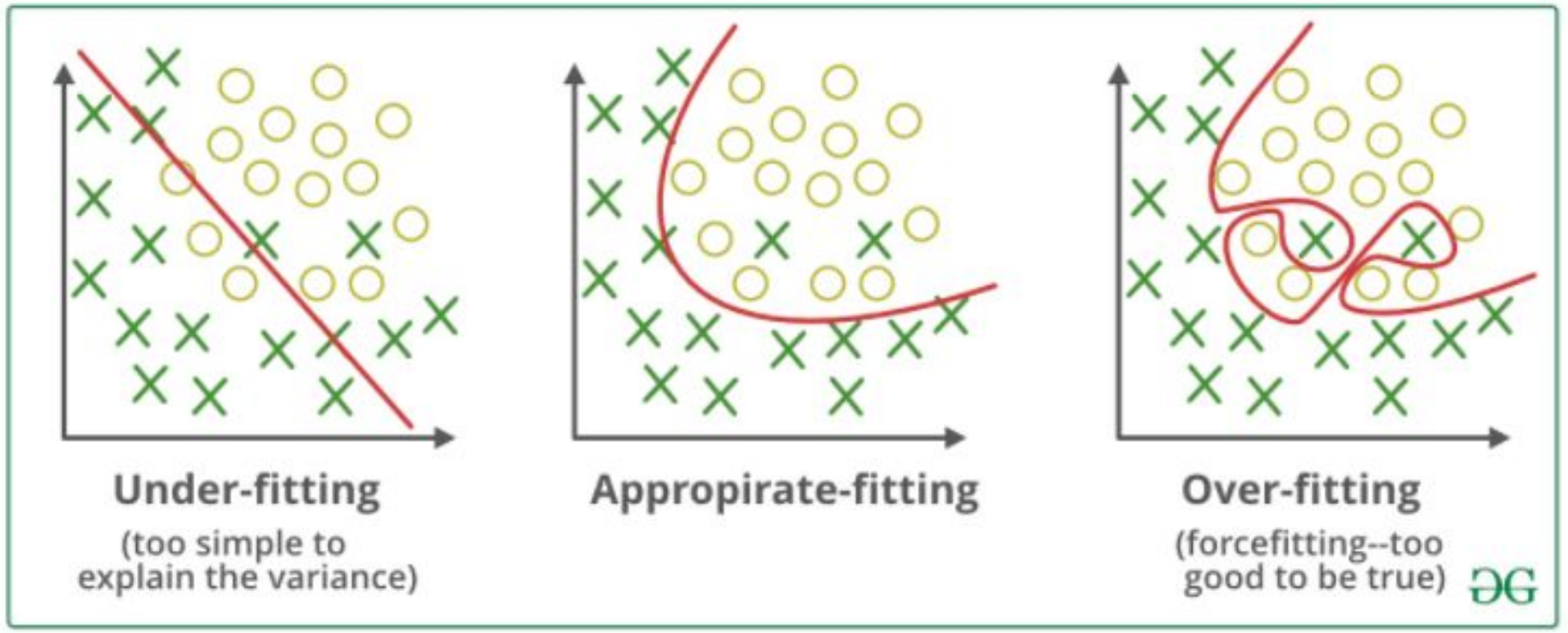




Regression overfitting and underfitting



Classification overfitting and underfitting



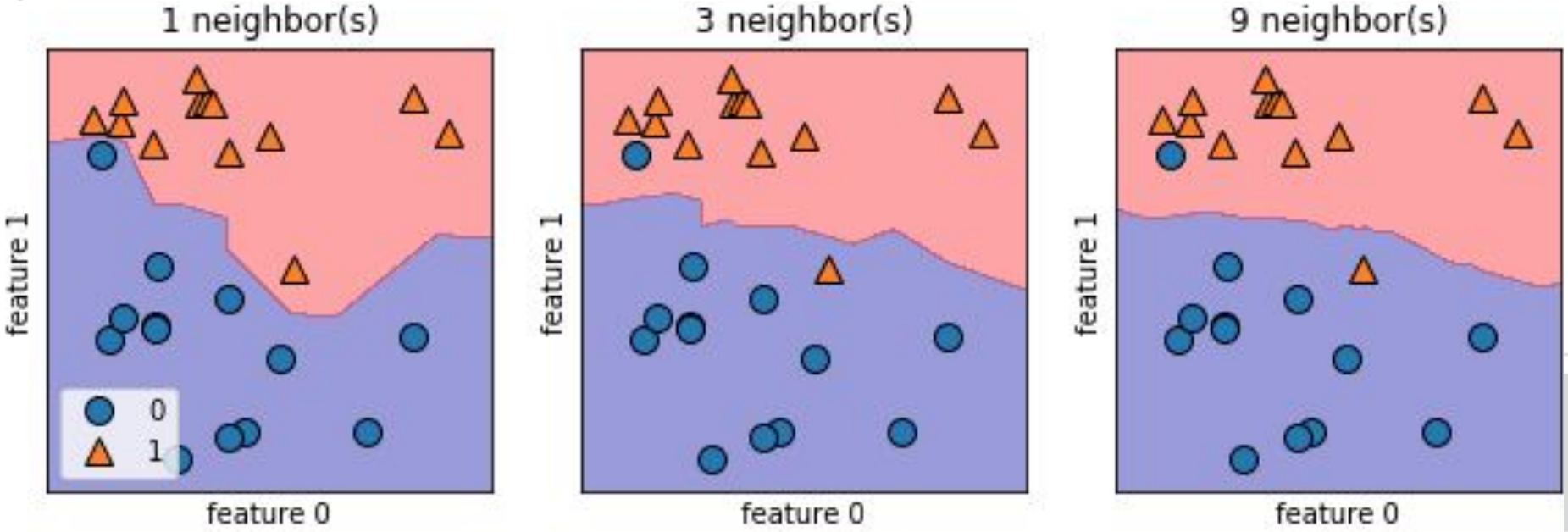
השפעת הפרמטר K

K קטן

קו החלטה עקלקל יותר – מודל יותר מורכב – הכללה פחותה

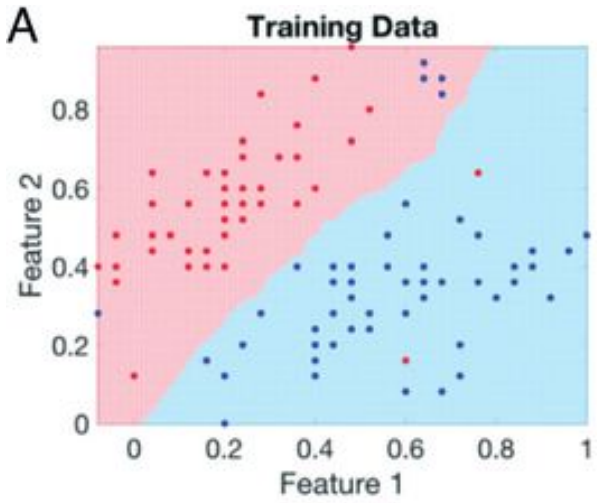
K גדול

קו החלטה חלק יותר – מודל פשוט יותר – הכללה טובה יותר



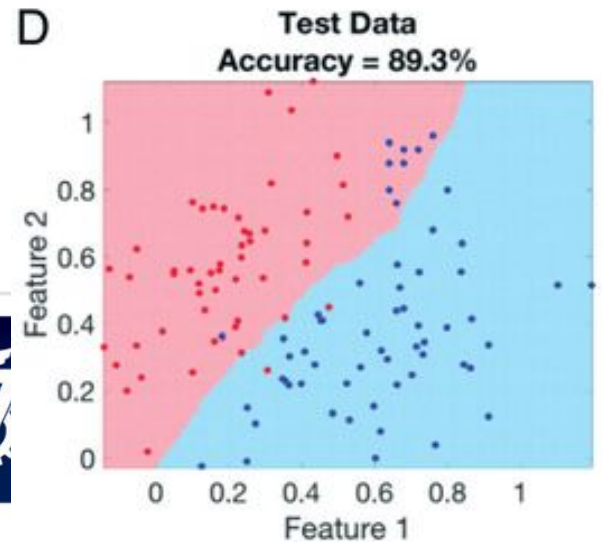
Overfit and underfit - KNN

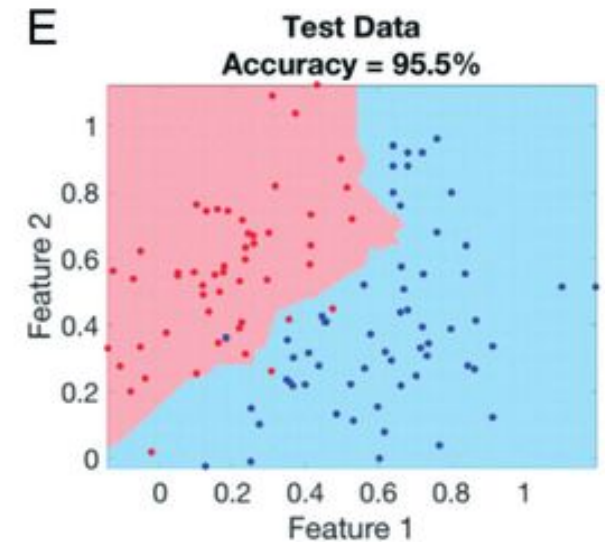
Large k – Small complexity

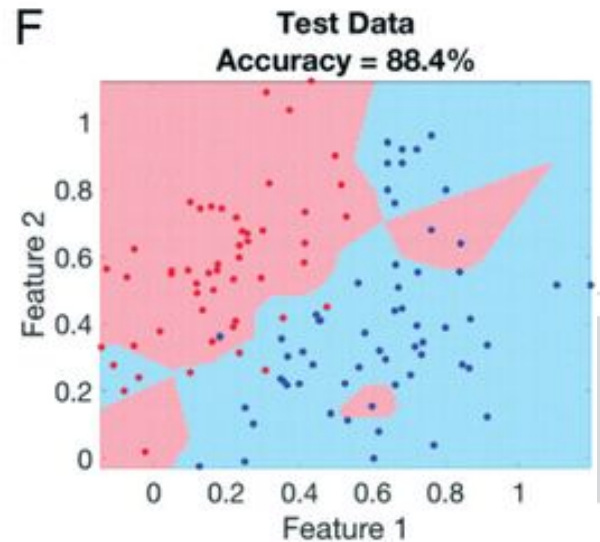


Medium k – Medium complexity

Small K – Large complexity







סיכום בטבלה

נרצה תמיד למדוד את הדיוק על נתוני האימון וגם נתוני המבחן. הדיוק על נתוני האימון מעיד עד כמה האלגוריתם למד. הדיוק על נתוני המבחן מעיד עד כמה ההכללה טובה. קיימות 3 אפשרויות המרוכזות בטבלה:

מצב	למידת חסר Underfitting	למידה טובה	למידת יתר overfitting
דיוק על נתוני האימון Train accuracy	נמוך	גבוה	גבוה מאד
דיוק על נתוני המבחן Test accuracy	נמוך	גבוה	נמוך

כיצד נמנעים מ overfitting

- הגדלת מספר דוגמאות האימון
- כיוון היפר-פרמטרים (K)
- יש דרכים נוספות, שילמדו בהמשך

שלבי אימון, ולידציה ומבחן

parameters - פרמטרים

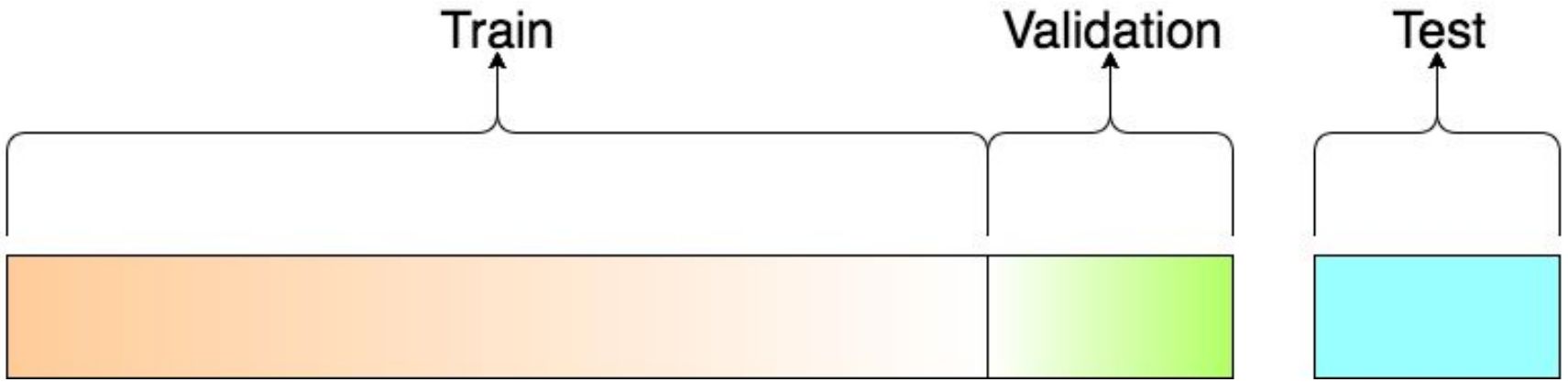
- פרמטרים נלמדים של האלגוריתם ושל תהליך הלמידה
- לדוגמא
  - w,b ברגרסיה לינארית
  - בעתיד הקרוב: w,b של SVM Linear
  - בעתיד הרחוק: w,b של שכבת נוירונים

היפר פרמטרים - hyper parameters

- פרמטרים מבניים של האלגוריתם ושל תהליך הלמידה
- לדוגמא
  - K ב-KNN
  - בעתיד הקרוב: סוג kernel ורגולריזציה באלגוריתם SVM
  - בעתיד הרחוק: מספר שכבות ברשת נוירונים

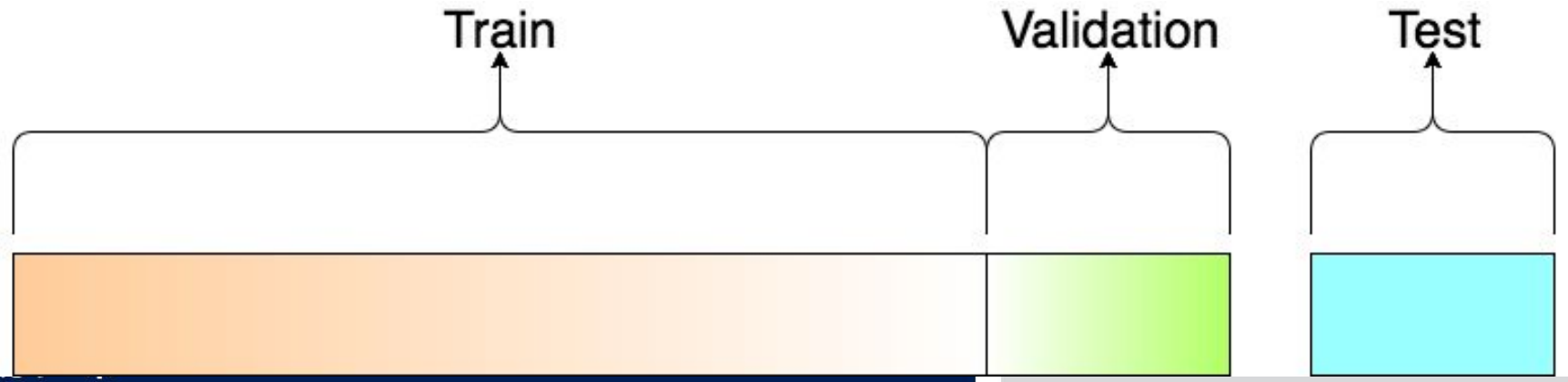
חלוקת מאגר הנתונים בתהליך הפיתוח

- Train – נתונים לאימון
- Validation – נתונים להערכת ביצועים בתהליך הפיתוח
- Test – נתונים להערכת הביצועים הסופיים של האלגוריתם הנבחר



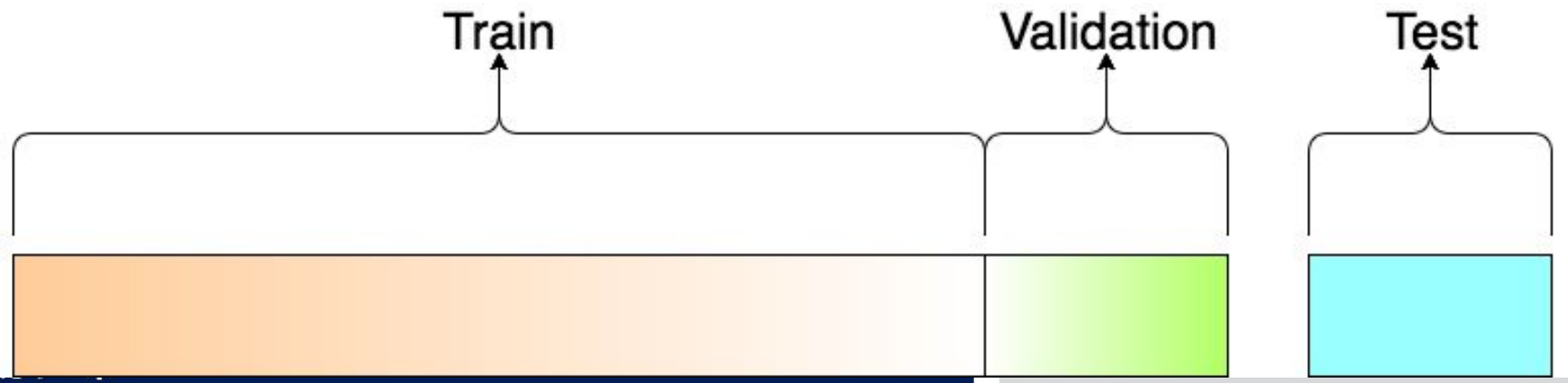
תהליך אימון מכונה לומדת
מה הבעיה כאן?

- בחירת היפר-פרמטרים
- אימון המסווג על Train
- בחינת ביצועים על Test
- בחירה סופית של היפר-פרמטרים
- בחינת ביצועים על Test



תהליך אימון מכונה לומדת
תהליך נכון

- בחירת היפר-פרמטרים
- אימון המסווג על Train
- בחינת ביצועים על Validation
- בחירה סופית של היפר-פרמטרים
- בחינת ביצועים על Test



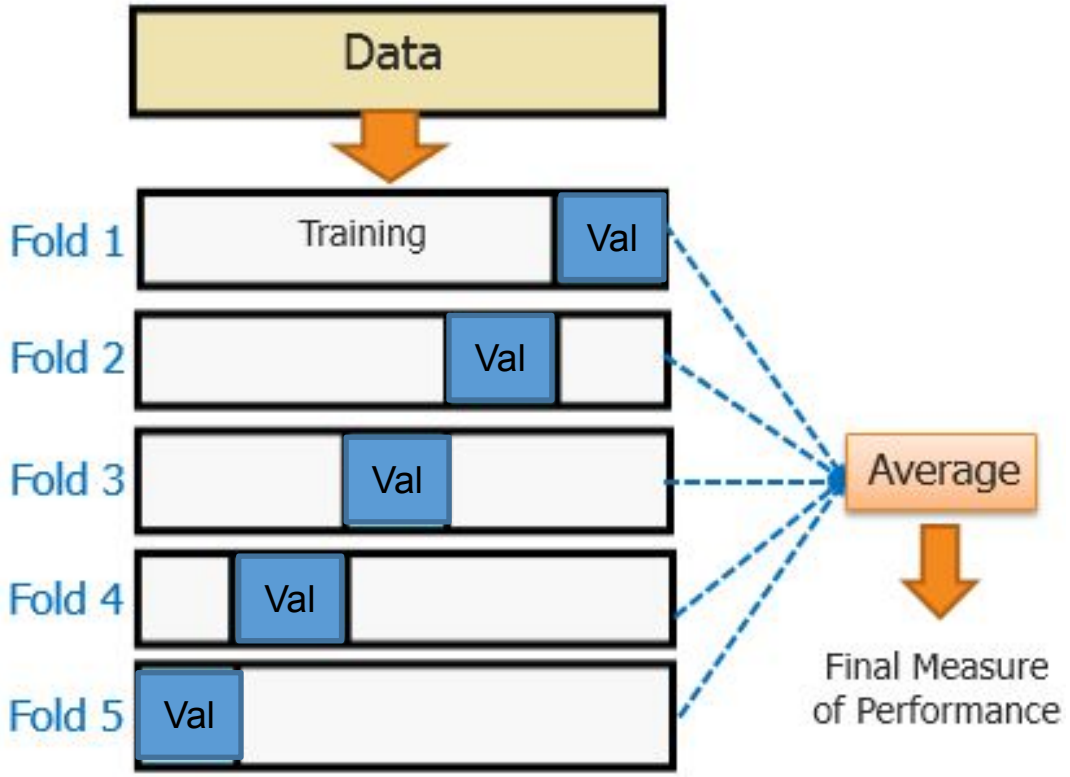
Train test split in sklearn

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

שימו לב: על מנת לחלק את הנתונים ל3 חלקים יש להריץ את פקודת החלוקה פעמיים:
חלוקה train+val, test
וחלוקה של train+val לtrain,val

שיטת cross validation

Cross validation



Cross validation in sklearn

https://scikit-learn.org/stable/modules/cross_validation.html

הטכניון
מכון טכנולוגי
לישראל

Hyper parameter tuning

imports

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.metrics import accuracy_score, ConfusionMatrixDisplay, classification_report
from sklearn.pipeline import make_pipeline
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
```

Method #1
using loops

Choosing best parameter K

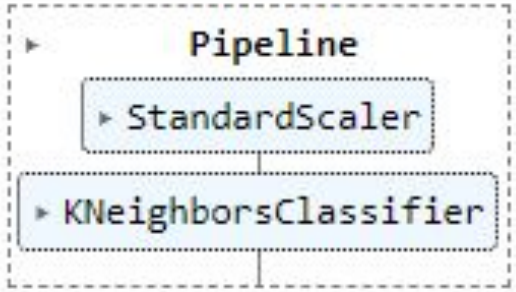
```python
#finding best_k
best_score = 0
for k in range(1,101,10):
  # create KNN classifier
  knn1 = make_pipeline(StandardScaler(), KNeighborsClassifier(n_neighbors = k))
  # cross validation score on train data
  scores = cross_val_score(knn1, X_train, y_train, cv=5)
  score = np.mean(scores)
  print(f"k={k}, mean score: {score}")
  if score > best_score:
    best_k = k
    best_score = score
print(f"best k:{best_k} best score: {best_score}")
```

```
k=1, mean score: 0.9698812019566738
k=11, mean score: 0.973654786862334
k=21, mean score: 0.9774283717679945
k=31, mean score: 0.9812019566736548
k=41, mean score: 0.9661076170510133
k=51, mean score: 0.9547868623340321
k=61, mean score: 0.9283717679944095
k=71, mean score: 0.9245981830887491
k=81, mean score: 0.856953179594689
k=91, mean score: 0.8381551362683437
best k:31 best score: 0.9812019566736548
```

Note: this is done right after Choosing X,y and Train-test split.
We normalize with classifier in pipeline, so that fit_transform is done on the train splits and transform is done on the validation splits within the cross validation process.

Choosing best parameter K

```
#chosing best_k
knn1 = make_pipeline(StandardScaler(), KNeighborsClassifier(n_neighbors = best_k))
knn1.fit(X_train,y_train)
```



Evaluate on test data

```
#predict on test samples
y_pred = knn1.predict(X_test)
y_pred.shape
```

```
(67,)
```

```
#accuracy on test samples
accuracy_score(y_true=y_test, y_pred = y_pred)
```

```
0.9701492537313433
```

Penguin Classification exercise – method #1

Penguin classification exercise 2.5

Method #2
using GridSearchCV

Choosing best parameter K

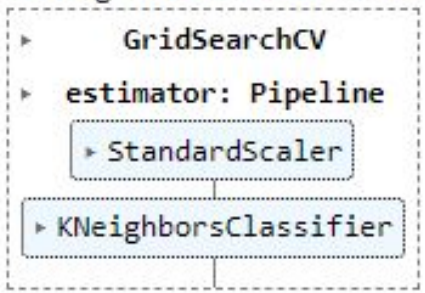
```
pipe = Pipeline([
        ('scale', StandardScaler()),
        ('knn', KNeighborsClassifier())
        ])
```

```
#Note format for dictionary keys: pipeName__parameterName
param_grid = {
    'knn__n_neighbors' :[k for k in range(1,101,10)]
}
param_grid
```

```
{'knn__n_neighbors': [1, 11, 21, 31, 41, 51, 61, 71, 81, 91]}
```

```
knn_clf = GridSearchCV(pipe, param_grid=param_grid, cv=5, verbose=1)
knn_clf.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```



Note: this is done right after Choosing X,y and Train-test split.
We normalize with classifier in pipeline, so that fit_transform is done on the train splits and transform is done on the validation splits within the cross validation process.

Choosing best parameter K

```
cv_results = pd.DataFrame(knn_clf.cv_results_)
cv_results[['param_knn__n_neighbors','mean_test_score','rank_test_score']].sort_values('rank_test_score')
```

	param_knn__n_neighbors	mean_test_score	rank_test_score
3	31	0.981202	1
2	21	0.977428	2
1	11	0.973655	3
0	1	0.969881	4
4	41	0.966108	5
5	51	0.954787	6
6	61	0.928372	7
7	71	0.924598	8
8	81	0.856953	9
9	91	0.838155	10

```
knn_clf.best_params_
```

```
{'knn__n_neighbors': 31}
```

Evaluate on test data

```
y_pred = knn_clf.predict(X_test)
y_pred.shape
```

```
(67,)
```

```
#accuracy on test samples
accuracy_score(y_true=y_test, y_pred = y_pred)
```

```
0.9701492537313433
```

Penguin Classification exercise – method #2

Penguin classification exercise 3.0

תודה על ההשתתפות