Instructions for the image exercise

TKO 3120 Machine Learning and Pattern Recognition

Petra Virjonen

pekavir@utu.fi

UNIVERSITY OF TURKU

Outline

- Exercise task in general
- Tools
- Data set
- Parts of the exercise
- Template
- Deadlines
- Grading
- Principal Component Analysis PCA
- K nearest neighbor algorithm (kNN)
- K-fold cross validation
- Nested cross validation

UNIVERSITY OF TURKU

Exercise task in general

- The task is to classify color images using supervised machine learning algorithms
  - 3 different classes of images, one label per image
- Gather a training set by extracting several features from the images
- Train different classifiers to identify the class of an image
  - K nearest neighbors algorithm (kNN)
  - Random forest
  - Multi-layer perceptron (MLP)
- Estimate the performance of each classifier
- Return your exercise as a working Python Jupyter notebook
- A notebook template including the link to the datasets, and instructions can be found in Moodle course page

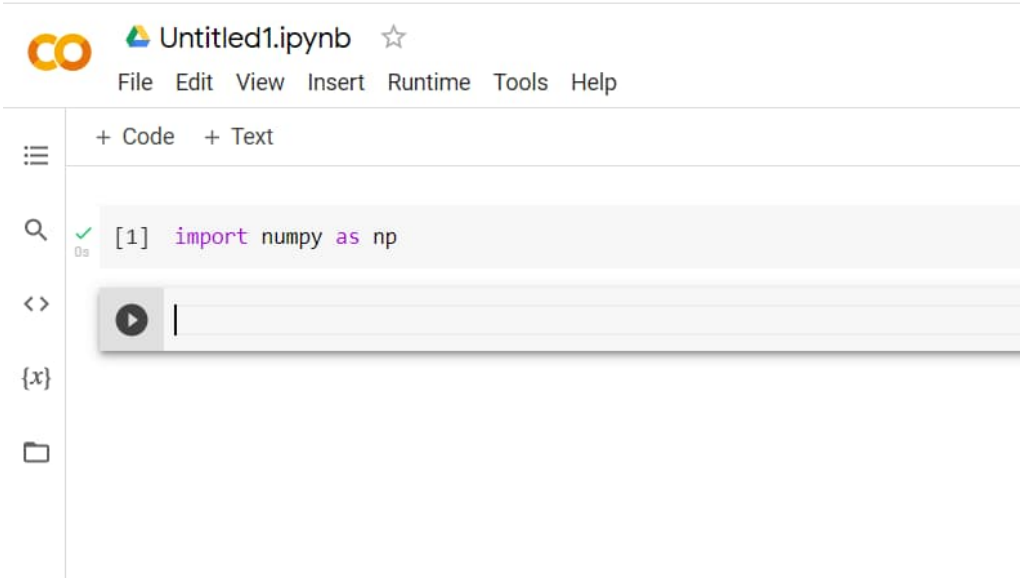
Anaconda

- Install Anaconda Distribution to your computer (Python 3)
  - Includes NumPy, pandas, Jupyter, scikit-learn, scikit-image, Matplotlib
    - https://www.anaconda.com/distribution/#download-section
  - You may use also other modules if needed (such as OpenCV)
- Tutorials
  - Jupyter Notebook
    - https://www.dataquest.io/blog/jupyter-notebook-tutorial/
  - Python 3
    - https://www.tutorialspoint.com/python3/
  - Scikit-learn
    - https://scikit-learn.org/stable/user_guide.html
  - Scikit-image
    - https://scikit-image.org/docs/dev/index.html
  - OpenCV
    - https://docs.opencv.org/4.x/index.html

UNIVERSITY OF TURKU

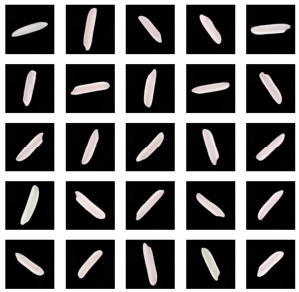
Google Colaboratory

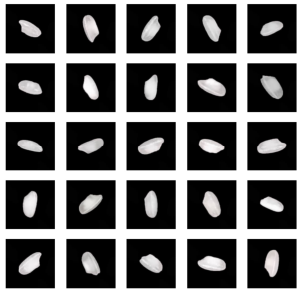
- Write and execute Python in your browser
- No configuration required
- Free access to GPUs
- Easy sharing
- https://colab.research.google.com/



Data set

- Task is to classify rice species from images
- Original article by İ. Çınar and M. Koklu*
- Original datasets include 15000 preprocessed images for five rice species (appr. 60 MB per rice species)
  - One rice in each image
  - Datasets can be downloaded from: https://www.muratkoklu.com/datasets/vtdhnd09.php
- Smaller sample deployed in this exercise
  - Three rice species
  - 100 images per rice species



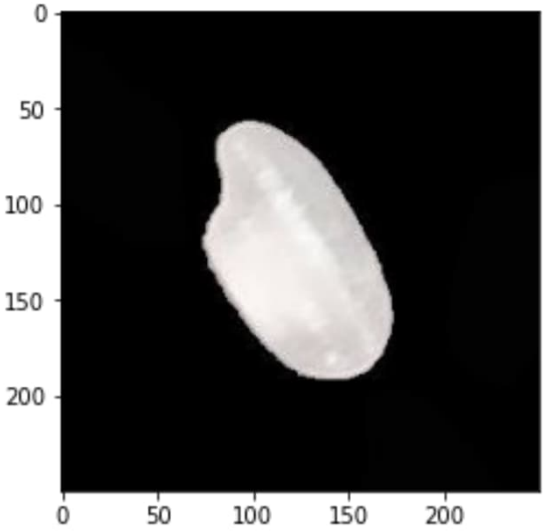


*İ. Çınar and M. Koklu. Identification of rice varieties using machine learning algorithms. Journal of Agricultural Sciences, 28(2):307–325, 2022. doi: 10.15832/ankutbd.862482

UNIVERSITY OF TURKU

Images

- RGB images
  - Stored as 3 dimensional arrays
    - [height, width, color channel]
  - Pixel values as 8-bit unsigned integers: [0,255]

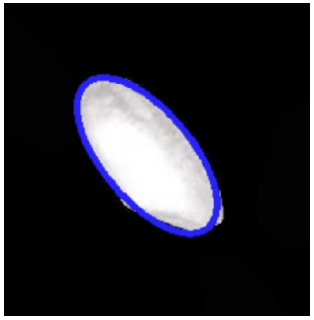



```
im = cv.imread(filename)
plt.imshow(im)
```

UNIVERSITY OF TURKU

Parts of the exercise

- **Part 1 (3 p)**
  - Perform preparations for the data
    - Import the sampled datasets
    - Determine the contour of each rice
  - Feature extraction
    - Fit an ellipse to each rice
    - Gather color/texture and shape features
- **Part 2 (4 p)**
  - Data exploration
  - Model selection
    - Select the best model for each classifier
- **Part 3 (5 p)**
  - Performance estimation of each model
  - Discussion
  - Introduction



UNIVERSITY OF TURKU

Template

- Write a clear *report*, understandable even for an unspecialized reader
  - use the template provided in Moodle course page
  - define shortly the concepts and explain the phases you use
  - use the Markdown feature of the notebook for larger explanations
- Return your output as a *working* Jupyter notebook
  - name your file using the universally unique identifier (run the cell only once and use this same identifier for each part)
  - do not include your name or any other identifiers

Deadlines

- Part 1 submission: Mon 6.2 at 23:59
- Part 1 grading: Mon 13.2 at 23:59
- Part 2 submission: Mon 20.2 at 23:59
- Part 2 grading: Mon 27.2 at 23:59
- Part 3 submission: Mon 6.3 at 23:59
- Part 3 grading: Sun 12.3 at 23:59

- Obey the deadlines! No extensions are granted.

UNIVERSITY OF TURKU

Grading

- The course grade consists of the exercise and the exam, total score 36 points
  - Exam has 5 questions, 6 points of each
  - Exercise gives 6 exam points
- From the exercise template, you can see how many exercise points can be acquired from each task. Exam points are given according to the table below:

  7 exercise points: **1 point**
  8 exercise points: **2 points**
  9 exercise points: **3 points**
  10 exercise points: **4 points**
  11 exercise points: **5 points**
  12 exercise points: **6 points**

- At least 7 exercise points are needed to pass the exercise
  - At least 1 exercise point from each part

UNIVERSITY OF TURKU

Peer grading

- Exercises are submitted in Moodle exercise area
- After each task deadline, an example solution and grading instructions are given
- Each student grades their own submission and one submission from anonymous peer according to the given instructions within one week
- Why peer grading?
  - Learn different ways to solve the exercises
  - Deeper understanding of the topic
  - Practice giving feedback

UNIVERSITY OF TURKU

Help!

- Three Q&A sessions are organized
  - Teacher is available for questions concerning the exercises and grading
  - 30.1 after lecture at 14:00 – 15:00 room 407B Honka (Agora 4th floor)
  - 13.2 after lecture at 14:00 – 15:00 room 407B Honka (Agora 4th floor)
  - 2.3 during lecture at 10:15 – 12:00 lecture hall XXI
- Discussion area at Moodle
- By e-mail pekavir@utu.fi

UNIVERSITY OF TURKU

General instructions

- Write easily readable code with comments
  - if you exploit some code from web, provide a reference
  - avoid redundant code! Exploit the relevant parts and modify the code for your purposes to produce only what you need
- It is ok to discuss about the assignment with a fellow student. But it is not ok to copy someone's work. Everyone should submit their own implementation. Identical submissions will lead to fail for both students.

1 Part 1

Read the original research article:

İ. Çınar and M. Koklu. Identification of rice varieties using machine learning algorithms. Journal of Agricultural Sciences, 28(2):307–325, 2022. doi: 10.15832/ankutbd.862482.

https://dergipark.org.tr/en/download/article-file/1513632

1.1 Introduction (1 p)

Will be written in Part 3

1.2 Preparations of the data (1 p)

Make three folders in your working folder: "notebooks", "data" and "training_data". Save this notebook in "notebooks" folder.

Perform preparations for the data

- import all the packages needed for this notebook in one cell
- import the images. Data can be found from (downloading starts as you press the link) https://www.muratkoklu.com/datasets/vtdhnd09.php
  - save the data folders "Arborio", "Basmati" and "Jasmine" in "data" folder
- take a random sample of 100 images from Arborio, Basmati and Jasmine rice species (i.e. 300 images in total)
- determine the contour of each rice (you can use e.g. *findContours* from OpenCV)
- plot one example image of each rice species, including the contour

1.3 Feature extraction (2 p)

Gather the feature data

Color features (15)

- Calculate the following color features for each image, including only the pixels within the contour (you can use e.g. *pointPolygonTest* from OpenCV)
  - Mean for each RGB color channel
  - Variance for each RGB color channel
  - Skewness for each RGB color channel
  - Kurtosis for each RGB color channel
  - Entropy for each RGB color channel

Dimension features (6)

- Fit an ellipse to the contour points (you can use e.g. *fitEllipse* from OpenCV)
- Plot one example image of each rice species including the fitted ellipse
- Calculate the following features for each image (for details, see the original article)
  - the major axis length the ellipse
  - the minor axis length of the ellipse
  - area inside the contour (you can use e.g. *contourArea* from OpenCV)
  - perimeter of the contour (you can use e.g. *arcLength* from OpenCV)
  - roundness
  - aspect ratio

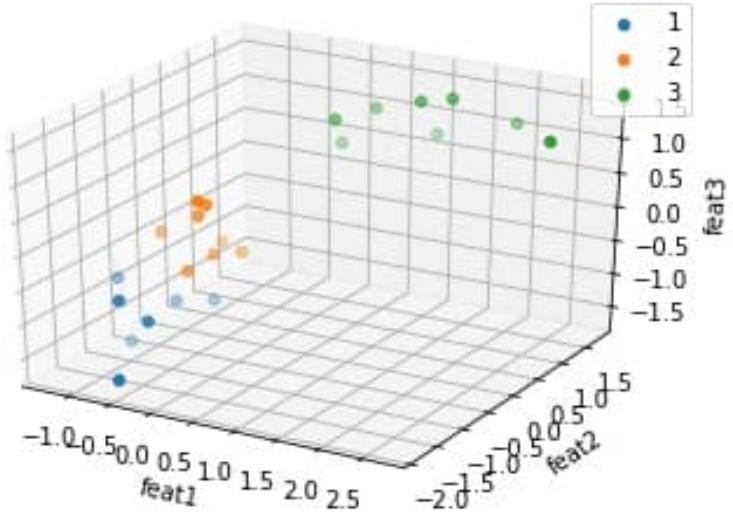
Gather all the features in one array or dataframe: one data point in one row, including all feature values in columns.
For each data point, include also information of the original image and the label (rice species). Save the data in "training_data" folder.

Principal Component Analysis PCA

- An unsupervised learning method
- A dimension reduction technique
  - Reduces data by transforming it from *n*-dimensional space to *q*-dimensional space (*q*<*n*)
  - Preserves as much as possible of the original variance of the data
    - → optimization problem
- The data is rotated so that
  - The first axis is the direction with the largest variation
  - Each axis is chosen so that it is orthogonal to the previous ones, and covers as much as possible of the remaining variation
- The resulting axes are
  - a linear combination of the original variables
  - called principal components
- PCA can be used for visualization using the first two principal components
  - Can be used to spot potential clusters
- PCA is sensitive to relative scaling of the original variables!

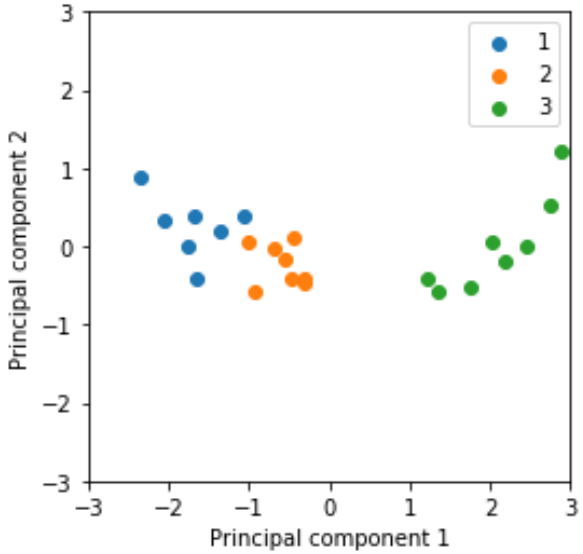
Simple example of PCA

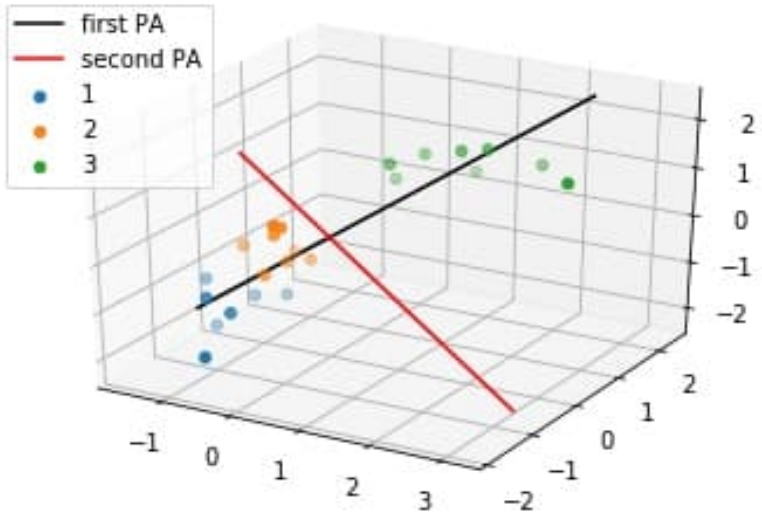
feat1	feat2	feat3	y
1.36	0.96	1.57	1
0.90	0.91	2.19	1
1.70	1.99	1.54	1
0.40	1.50	2.30	1
1.24	1.72	1.58	1
0.85	1.21	0.37	1
1.44	0.20	0.14	1
1.63	1.77	5.37	2
0.82	1.97	3.76	2
1.63	2.09	3.14	2
1.49	1.95	4.63	2
1.94	2.33	3.14	2
1.53	1.64	2.89	2
1.50	2.46	3.23	2
1.65	1.93	5.12	2
3.02	4.01	7.39	3
4.99	4.62	7.06	3
2.66	3.70	6.26	3
6.12	3.75	7.52	3
3.83	4.36	6.46	3
2.74	3.43	7.34	3
3.84	3.92	7.99	3
4.26	4.08	8.11	3



UNIVERSITY OF TURKU

PCA example cont.





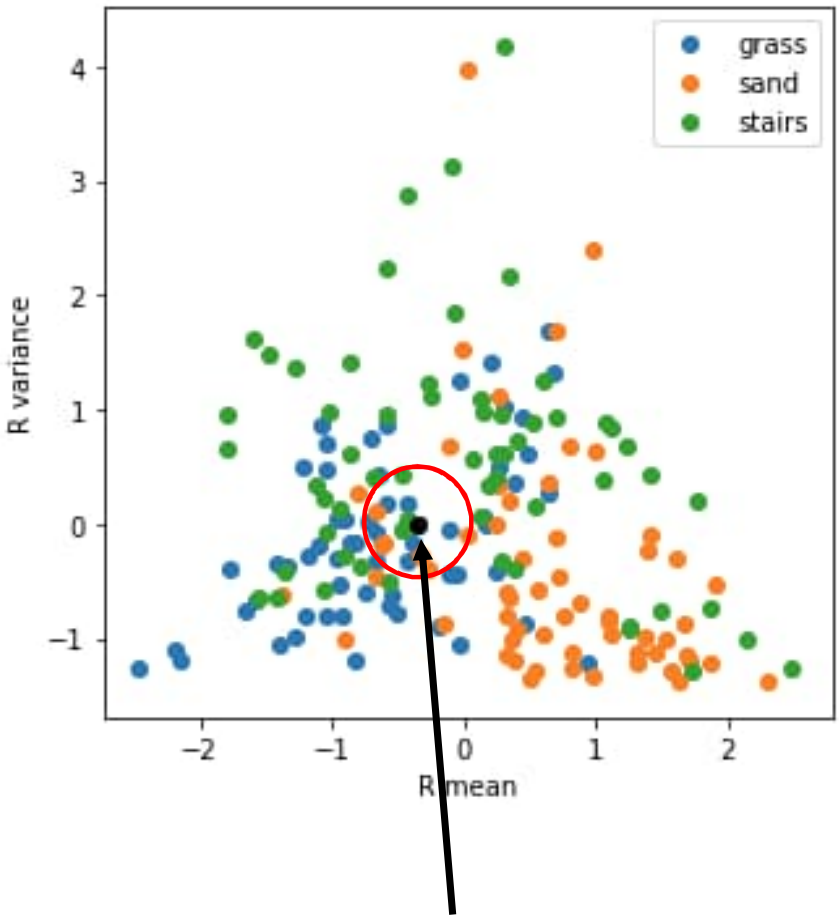
- PCA from sklearn.decomposition

UNIVERSITY OF TURKU

K nearest neighbors algorithm

- Supervised machine learning algorithm
- Does not learn a discriminative function but uses the whole training data instead
- Finds the k nearest neighbors from the training data, which are most similar to the input
- Distance measure
  - Euclidean distance most common
  - Manhattan, Minkowski, Hamming other options
- Can be used for both regression and classification
  - Regression: output is the mean of the values of the k nearest neighbors
  - Classification: output is the most common class within the k nearest neighbors
- Majority voting: more frequent classes dominate
    - → weighting can be applied according to the distance from the input to the k nearest neighbors
- The best k depends on the data
  - Hyperparameter k can be optimized in model selection

Example of kNN



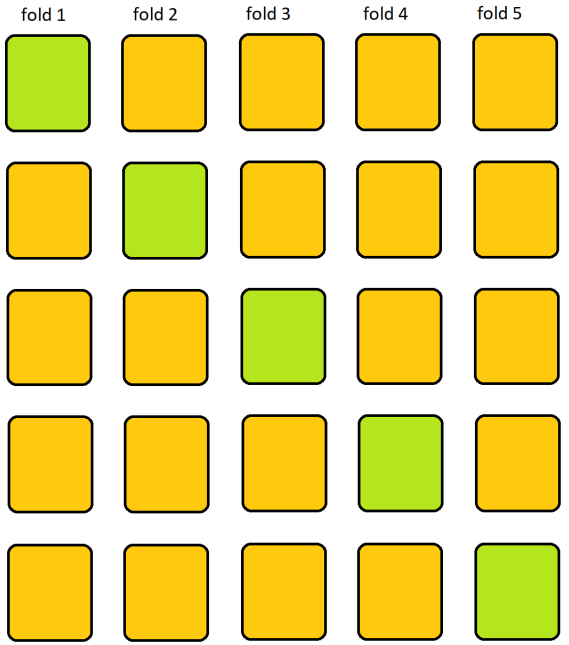

```
k= 1
Predicted class: 2
Indices of the nearest neighbors: [155]
Classes of the nearest neighbors: [2]
k= 2
Predicted class: 2
Indices of the nearest neighbors: [155 134]
Classes of the nearest neighbors: [2 2]
k= 3
Predicted class: 2
Indices of the nearest neighbors: [155 134  53]
Classes of the nearest neighbors: [2 2 0]
k= 4
Predicted class: 2
Indices of the nearest neighbors: [155 134  53 158]
Classes of the nearest neighbors: [2 2 0 2]
k= 5
Predicted class: 2
Indices of the nearest neighbors: [155 134  53 158  17]
Classes of the nearest neighbors: [2 2 0 2 0]
k= 6
Predicted class: 0
Indices of the nearest neighbors: [155 134  53 158  17  57]
Classes of the nearest neighbors: [2 2 0 2 0 0]
k= 7
Predicted class: 0
Indices of the nearest neighbors: [155 134  53 158  17  57  47]
Classes of the nearest neighbors: [2 2 0 2 0 0 0]
k= 8
Predicted class: 0
Indices of the nearest neighbors: [155 134  53 158  17  57  47 117]
Classes of the nearest neighbors: [2 2 0 2 0 0 0 1]
k= 9
Predicted class: 0
Indices of the nearest neighbors: [155 134  53 158  17  57  47 117 111]
Classes of the nearest neighbors: [2 2 0 2 0 0 0 1 1]
k= 10
Predicted class: 0
Indices of the nearest neighbors: [155 134  53 158  17  57  47 117 111  26]
Classes of the nearest neighbors: [2 2 0 2 0 0 0 1 1 0]
```

	r_mean	r_var
0	-1.273651	-0.981500
1	-0.030270	1.260521
2	-2.138505	-1.199752
3	-0.499541	-0.782923
4	-0.346548	-0.011617
5	-0.721096	0.008961
6	-0.823809	-0.169204
7	-0.111949	-0.445735
8	-0.867292	-0.159891
9	-1.399852	-1.049315
10	-1.048305	-0.796250
11	-0.645169	0.418593
12	-0.659911	-0.312166
13	-1.345162	-0.364733
14	0.264382	0.502165
15	0.374300	0.359103
16	-1.654791	-0.762096
17	-1.107411	-0.190239
18	-0.381914	-0.163425
19	0.205536	1.400269
20	-0.041301	-1.051041
21	-1.198540	-0.811814
22	-1.764977	-0.389903
23	-2.187270	-1.093669
24	-0.432363	-0.317261

K-fold cross validation

- A single split to test/training sets may lead to unstable results
- A procedure to estimate a model performance with unseen data (data which has not taken part in the training of the model)
- Data is split into subsets called folds
  - Each fold has approximately the same size
  - Each fold is used once as a test data while the remaining folds are used for model training
  - At each round, a performance estimation is calculated
  - The total performance is calculated as the mean performance of all rounds
- The performance estimation may still vary depending on the split
  - Repeated K-fold: K-fold repeated several times with different randomization in each repetition
  - sklearn.model_selection.RepeatedKFold

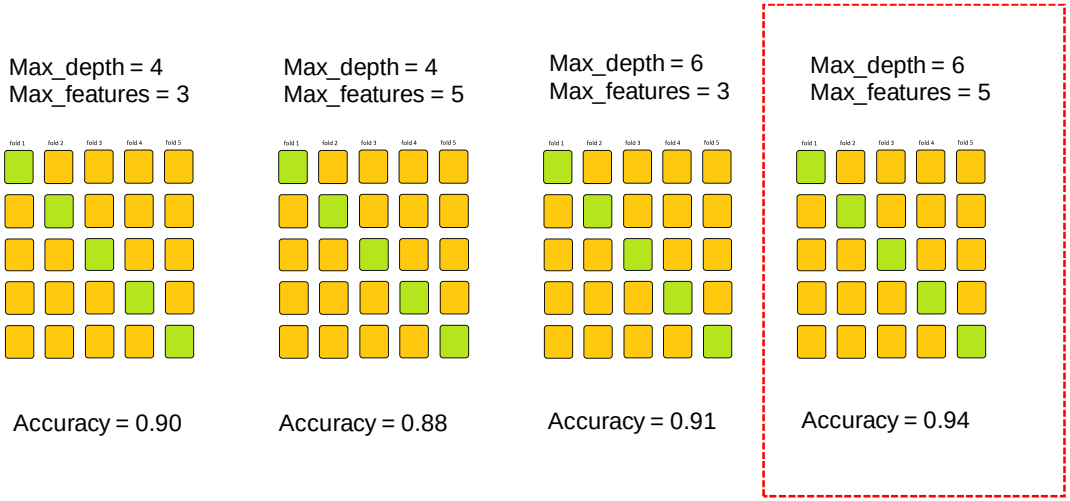
5-fold CV



UNIVERSITY OF TURKU

Model selection using K-fold CV

- For each hyperparameter combination, a performance estimation can be calculated using K-fold CV
- The model with the best performance estimation value is selected
- sklearn.model_selection.GridSearchCV
- Example: model random forest, hyperparameters max_depth: {4, 6}, max_features: {3, 5}



UNIVERSITY OF TURKU

Performance estimation using nested CV

- When the model is selected using all the data, there is no data left for testing the selected model
- In nested cross validation, cross validation is used to estimate the performance of the selected model
  - The model is selected with CV in the inner loop
  - The selected model performance is estimated in the outer loop
  - The total performance is calculated as the mean performance of all outer rounds
  - Different hyperparameters may be selected within the rounds
- **Gives an estimation of how a model, selected in this particular way, would perform with unseen data**