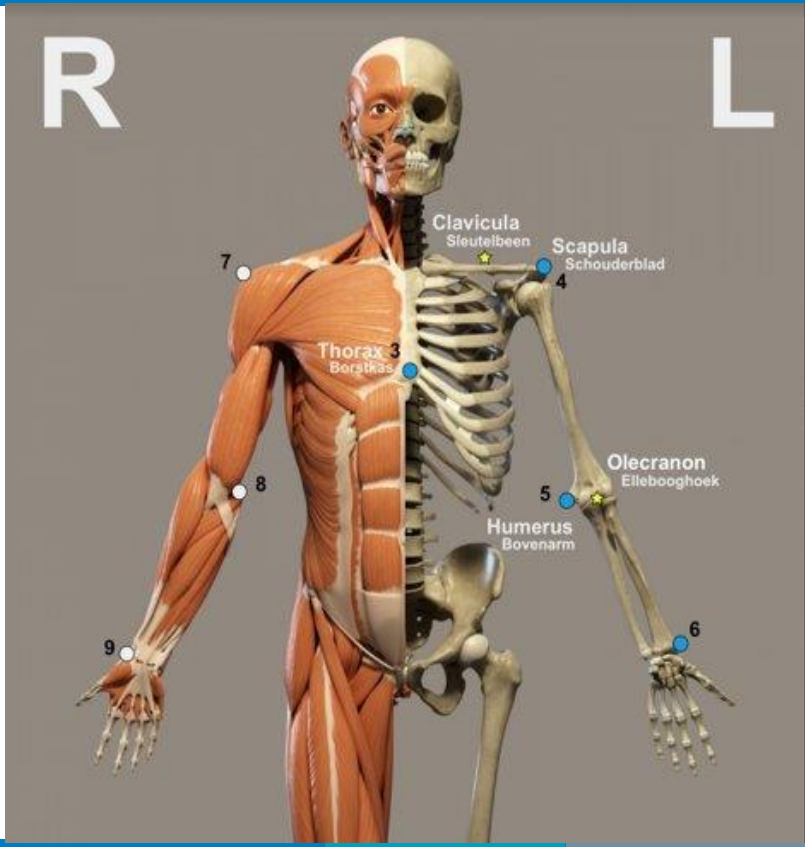
Leids Universitair Medisch Centrum

THE HAGUE UNIVERSITY OF APPLIED SCIENCES

Ortho Eyes

Dr. Tony Andrioli
Raphael, Brice, Eddie, Hassan, Arjun, Lennart, Rachelle



Leids Universitair
Medisch Centrum

Data processing

Hassan Ali

Data processing for Machine learning model

General Steps:

- How did we get 650 columns?
- Read in a csv file
- Read in patient folder with multiple exercises and save path of all exercises
- Read in one category folder ,multiple exercises
- folder with multiple categories and saving data
- cross joining exercises and creating 650 columns.

The data

Category_folder

Catagory_1
Catagory_2
Catagory_3

Patientgroup

1
2
3
4
5
•
•
•
25
26
27
28
29
30

Patientfolder (csvfiles)

m
AB1
AB2
AF1
AF2
EH1
EH2
EL1
EL2
RF1
RF2

Read in a csv file: Class exercises

- Read data from csv files and save the data as a pandas Dataframe
  - data = pd.read_csv("filename.csv")
- Rename column names
  - data.rename(columns= colnames)
- Save data in numpy array
  - data = pd.read_csv("filename.csv")
- Getting metadata from exercises (exercises type, category, etc)
  - os.path.split(exercisepath)
  - ‘CODE\data2.0\Catagory_1\1\AB1.csv’

Get frames (5 rows only)

dataframe.shape = 100 raws by 27 columns
Dataframe.size = 100x27
Raws = dataframe.size/len(columns) => 2700/27 = 100
Frames = 5,
Length dataframe = raws,
Steps_between frames = $\frac{rows}{Frames}$

$Formula =$ Steps_between frames * step_number

Voorbeeld: size = 96x27, get frames (1-5):

Steps_between frames = $\frac{96}{5}$ = 19

Voorbeeld get frames

Step: No:	Calculation	Frames
1	1*19	19
2	2*19	38
3	3*19	57
4	4*19	76
5	5*19	95

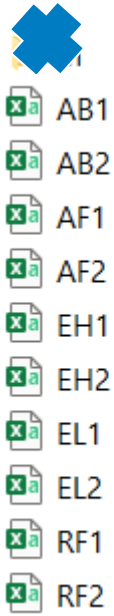
```python
def total_rows(self):
    return int(self.data.size / len(self.data.columns))

def get_frames(self):
    frames = []
    total_rows = self.total_rows() - 1
    for index in range(1, config.frames_counts + 1):
        frames.append(int((total_rows/ config.frames_counts) * index))
    return frames
```

dataframe_2 = dataframe.iloc[get_frames()]

Frames	thorax_r_x	thorax_r_y	thorax_r_z	.	humerus_l_y	humerus_l_z	ellebooghoek_l
19	4,281002	-4,902092	-3,568476	.	49,40454	1,703396	361,781
38	3,983373	-5,070366	-6,898982	.	140,0478	-14,25236	361,6263
57	7,701193	-4,54045	-7,870923	.	133,0988	-11,46238	369,8746
76	7,036899	-5,193945	-4,345288	.	43,63235	-7,319636	361,6309
95	7,796869	-3,205833	-0,6876775	.	15,97036	50,46438	403,7818

Patient

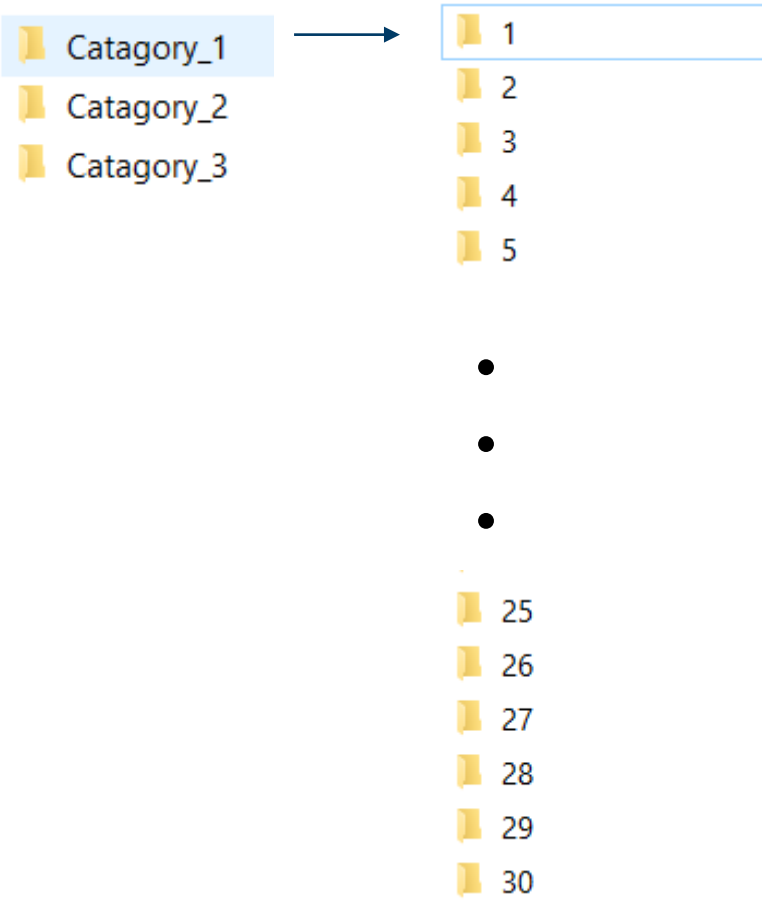
Patient Catagory_1\1



- Find if it's a csv file: endswith '.csv'
- csvfile = os.path.join(self.path, filename)
- Initiate class Exercises with path as csvfile.
- Create a list of reverences of all exercises with the path to an individual exercise

Patientgroup

Patientgroup



- Find name folder = os.listdir (path)
- Patientpath = os.path.join(self.path, name)
- Initiate class patient with path as Patientpath.
- Create a list of reverences of all patients from a category with the path to an individual patient

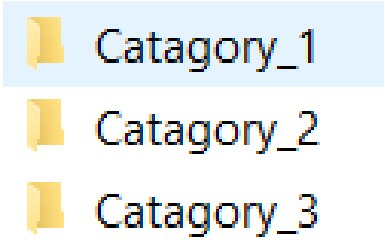
What do we have!

- ✓ Read in a csv of an exercise :- 5 frames
- ✓ A reference to all exercises of a patient
- ✓ A reference to all patients from a category

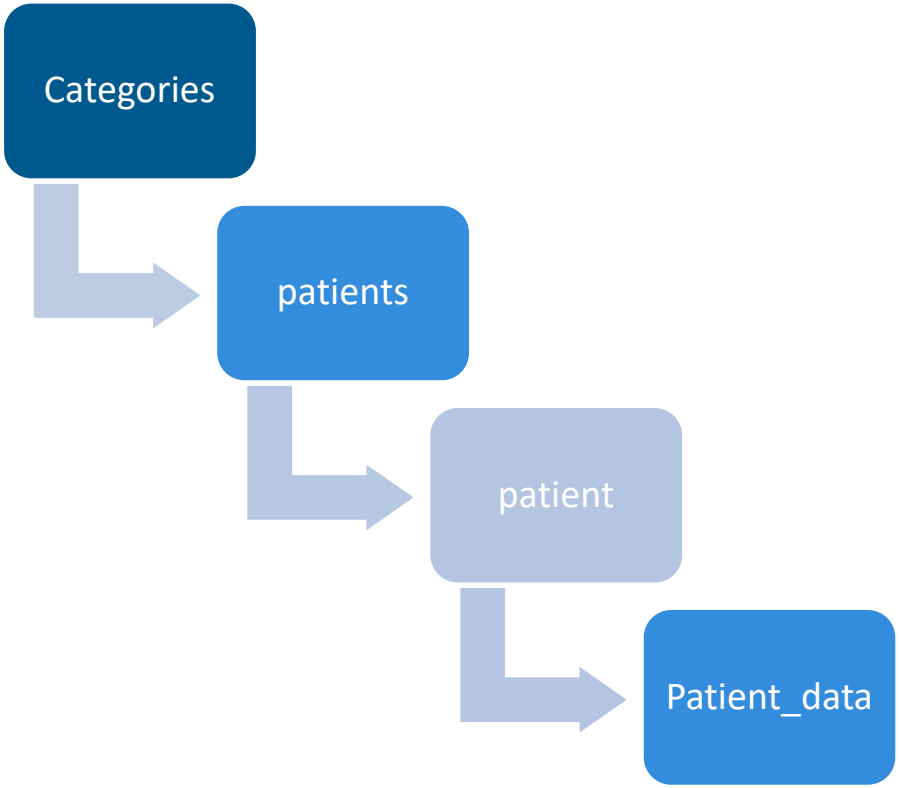
Patientgroups

- Create an empty list of patientgrougs
- Get grouppath by looping through folder containing patietgroups and get the groupID.
- Initiate class patientgroup with path as patientID.
- Create a list of reverences of all patientgroups with the path to an all patients
- To access an exercises is then as easy as:
  - patient_group[0].patients[0].exercises[0].dataframe

```
CODE\\data2.0\\Catagory_{groupid}
```



Create a patient data



- Loop through each and get the patient data and sort it out.
- Use the key exercise.exercisegroup to get the name of the exercise

AB1	**AB2**
AF1	**AF2**
EL1	**EL2**
RF1	**RF2**
EH1	**EH2**

Cross-joining exercises

- Generate more data by creating possible combinations for a patient.
- Example in test.py
- Patient_data

AB	AB1	AB2
AF	AF1	AF2

- Restults = list(itertools.product (patient_data[‘AB’], Patient_data[‘AF’]))
- [('AB1' 'AF1') ('AB1' 'AF2') ('AB2' 'AF1') ('AB2' 'AF2’)]

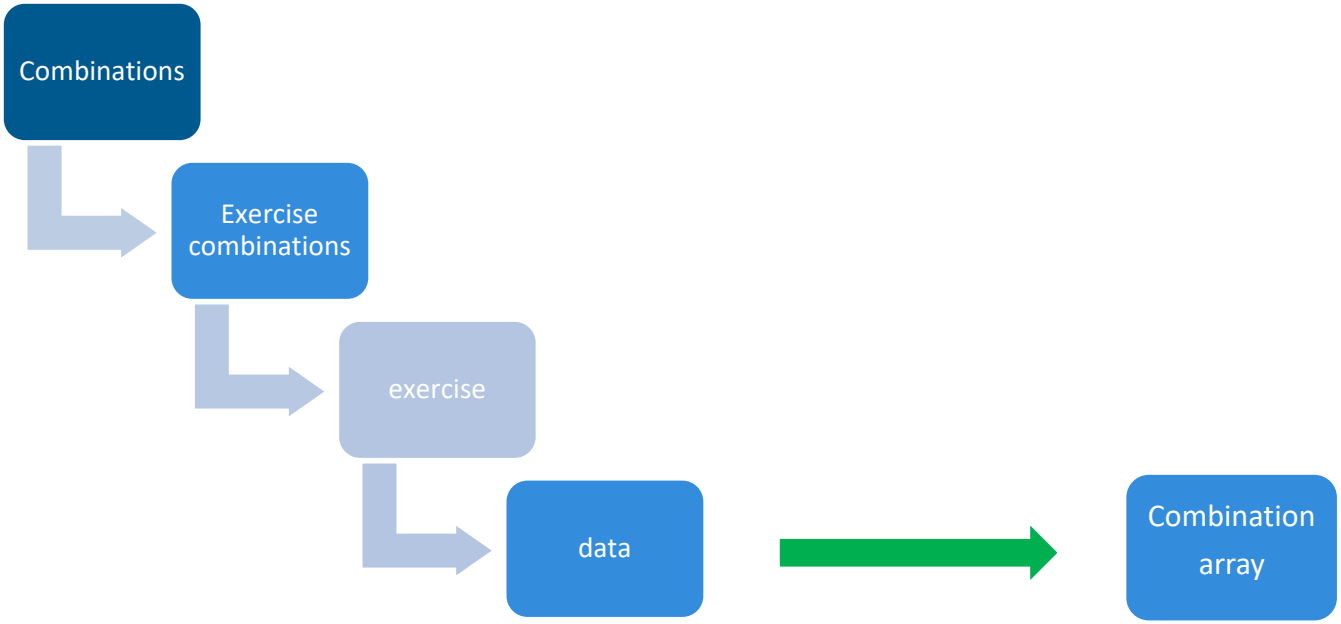
- Now results data is 2x more than the original patient_data
- Working with references to files instead of dataframes: see code

Splitting test & train data

- Randomly select patients by their id's for test in data
- If patient id is in test then it's patient_data from results goes to the test data
- Else it's a train data. See code

Getting the 650 columns!

- Loop through the patient data and get the data of each exercises => save this in an empty array
- Save the array in a combination_array with 650 columns => see code



Getting the group of each exercise combinations

Getting the group number:
This will be the answer for our model, de y

```
np_combination_array = np.vstack ([np_combination_array, data])
np_indicator_array = np.append(np_indicator_array, exercise_combination[0].patiengroup)
```

Questions & feedback moment