About the project

The end goal of this project is to classify patients with high protein concentration in urine and the healthy group based on SERS (Surface Enchanced Raman Spectroscopy) spectral data and biomedical data.
This project is to be released as a research paper later in 2022 or 2023. Some information is not fully shown here as a result.

The project is divided into several Jupyter notebooks with the following names: 1) Import raw urine spectra (part 1) 2) Spectra processing (part 2) 3) Classification of patients (part 3)

Author of all codes: Sultan Aitekenov, sultanaitekenov@gmail.com

Part of the upcoming abstract: Excessive protein excretion in human urine is an early and sensitive marker of diabetic nephropathy, primary and secondary renal disease. Kidney problems, particularly chronic kidney disease, remain among the few growing causes of mortality in the world. Therefore, it is important to develop efficient, expressive, and low-cost method for protein determination. Surface enhanced Raman spectroscopy (SERS) methods are potential candidates to achieve those criteria. In this paper, the SERS methods was developed to distinguish patients with proteinuria and the healthy group. Commercial gold nanoparticles with the diameter of 60 nm and 100 nm, and silver nanoparticles with the diameter of 100 nm were employed. Silver, gold, silicon and test slides covered with aluminium tape were utilized as substrates. Obtained spectra were analysed with several machine learning algorithms coupled with the PCA, ROC curve, and cross-validation methods.

The end goal is to create a dictionary file that contains information about each patient (keys) and their respective spectra in the pandas' object type (values).# Processe spectra, create figures and save output into a file This script runs for 371 sec.

Spectra processing (part 2)

This part of the project is dedicated to spectra processing. In other words, this part is the feature extraction part.

Raman spectra contain fluorescence background, spikes and noises. Several algorithms, such as the asymetric least squares, the median filter, the Savitsky Golay filter and the normalization by area, are used here. Then the resulting processed spectra are plotted.

Loads all spectra and raman shift

In [1]:
```
# import information from pickle file
import pickle
```

```
file_check=open("raw_urine_spectra.pkl","rb")
# # content of raw urine spectra
# raw_urine_spectra={"Ag_100nm_AgNPs": Ag_100nm_AgNPs,
#                    "Ag_100nm_AuNPs": Ag_100nm_AuNPs,
#                    "Al_tape_60nm_AuNPs": Al_tape_60nm_AuNPs,
#                    "Al_tape_100nm_AuNPs": Al_tape_100nm_AuNPs,
#                    "Au_60nm_AuNPs": Au_60nm_AuNPs,
#                    "Au_100nm_AuNPs": Au_100nm_AuNPs,
#                    "Si_60nm_AuNPs": Si_60nm_AuNPs}
raw_urine_spectra=pickle.load(file_check)
```

In [2]:
```
# import raman shift data
# Si_60nm_AuNPs is not included though
import pandas as pd
import numpy as np
raman_shift_400_1800=np.array(pd.read_csv('raman_shift_400_1800.csv', header=None))
wave = raman_shift_400_1800
```

Functions to process spectra

In [3]:
```
# ALS algorithm to remove fluorescence
# taken from https://qiita.com/h398qy988q5/items/cbf034b13c3cf0bb7761
def baseline_als(y, lam, p, niter=10):
    from scipy import sparse
    from scipy.sparse.linalg import spsolve
    L = len(y)
    D = sparse.diags([1,-2,1],[0,-1,-2], shape=(L,L-2))
    w = np.ones(L)
    for i in range(niter):
        W = sparse.spdiags(w, 0, L, L)
        Z = W + lam * D.dot(D.transpose())
        z = spsolve(Z, w*y)
        w = p * (y > z) + (1-p) * (y < z)
    return z
```

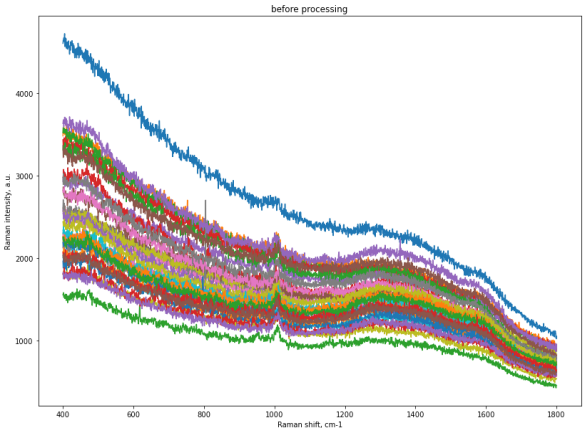
In [4]:
```
# process individual spectra by taking out the fluorescence, and applying filters
def raman_process_each_spectra(input_dict):
    import numpy as np
    import copy
    from scipy.ndimage import median_filter
    from scipy.signal import savgol_filter
    # output dictionary
    output_dict = copy.deepcopy(input_dict)

    # process individual spectra
    for i in input_dict.keys():
        # loop over individual spectra for a given patient
        for j in range(0, len(input_dict[i])):
            # print(output_dict[i][j,:])
            spectrum = input_dict[i][j,:]
            baseline = baseline_als(spectrum, 100000, 0.001)
            corrected = spectrum - baseline
            corrected2 = median_filter(corrected, size=10)
            corrected3 = savgol_filter(corrected2, 21, 2)
            output_dict[i][j,:] = corrected3

    return output_dict
```

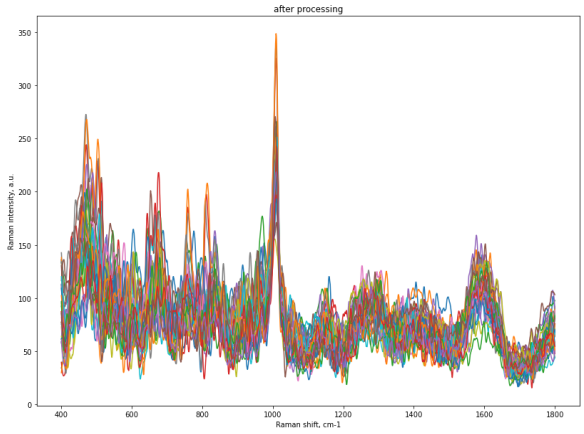
Perform calculations

In [5]:
```
import copy

processed_urine_spectra_before_combined = copy.deepcopy(raw_urine_spectra)

# cycles over keys and saves into a deepcopy output
for key in raw_urine_spectra.keys():
    processed_urine_spectra_before_combined[key] = raman_process_each_spectra(raw_urir
```

Create plots for an individual patient

In [6]:
```
# choose experimental set and a patient's ID to display
exp_set='Al_tape_60nm_AuNPs'
patient_ID = 3

import matplotlib.pyplot as plt
plt.figure(figsize=(30,10))
plt.subplot(1,2,1)
for i in range(0,len(raw_urine_spectra[exp_set][patient_ID])):
    plt.plot(wave[0], raw_urine_spectra[exp_set][patient_ID][i])
    plt.title('before processing')
    plt.xlabel('Raman shift, cm-1')
plt.ylabel('Raman intensity, a.u.')
plt.subplot(1,2,2)
for i in range(0,len(processed_urine_spectra_before_combined[exp_set][patient_ID])):
    plt.plot(wave[0], processed_urine_spectra_before_combined[exp_set][patient_ID][i])
    plt.title('after processing')
    plt.xlabel('Raman shift, cm-1')
    plt.ylabel('Raman intensity, a.u.')
```





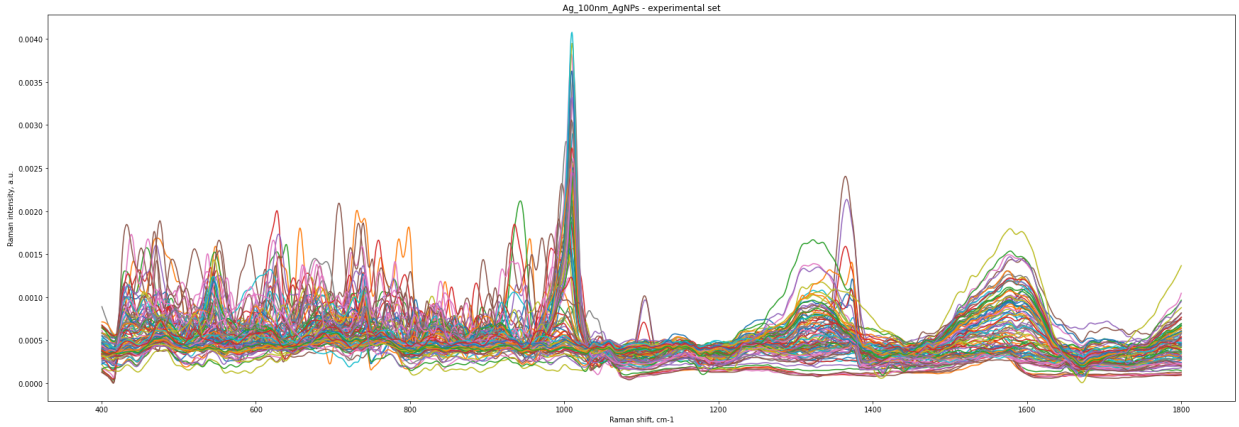
Combine individual spectrum for each patient

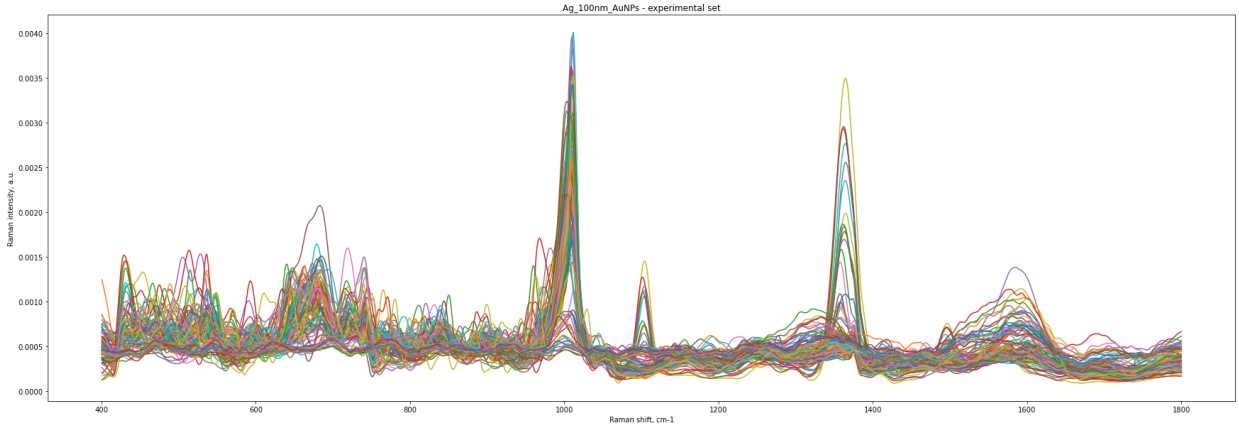
In [7]:
```
# Combine and normalize individual spectrum for each patient
from sklearn.preprocessing import maxabs_scale
from sklearn.preprocessing import minmax_scale

# create a deep copy with a final output
processed_urine_spectra=copy.deepcopy(processed_urine_spectra_before_combined)
```

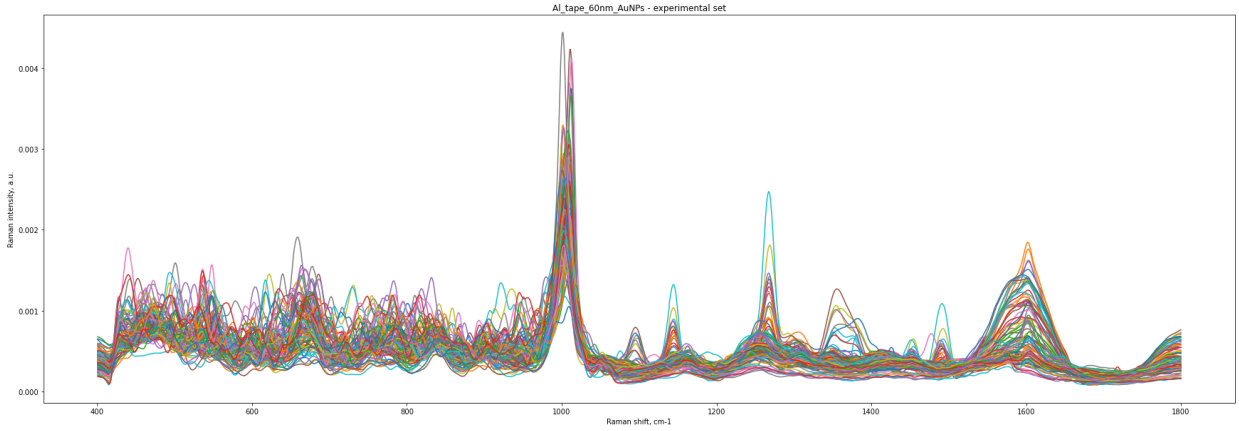
```
# Combine and normalize individual spectrum for each patient
for key_set in processed_urine_spectra_before_combined.keys():
    for key_ID in processed_urine_spectra_before_combined[key_set].keys():
        # combine by taking mean out of individual spectra for each patient
        combined = np.mean(processed_urine_spectra_before_combined[key_set][key_ID], a
        # normalize by unit area
        processed_urine_spectra[key_set][key_ID] = combined / combined.sum(axis=0)
```

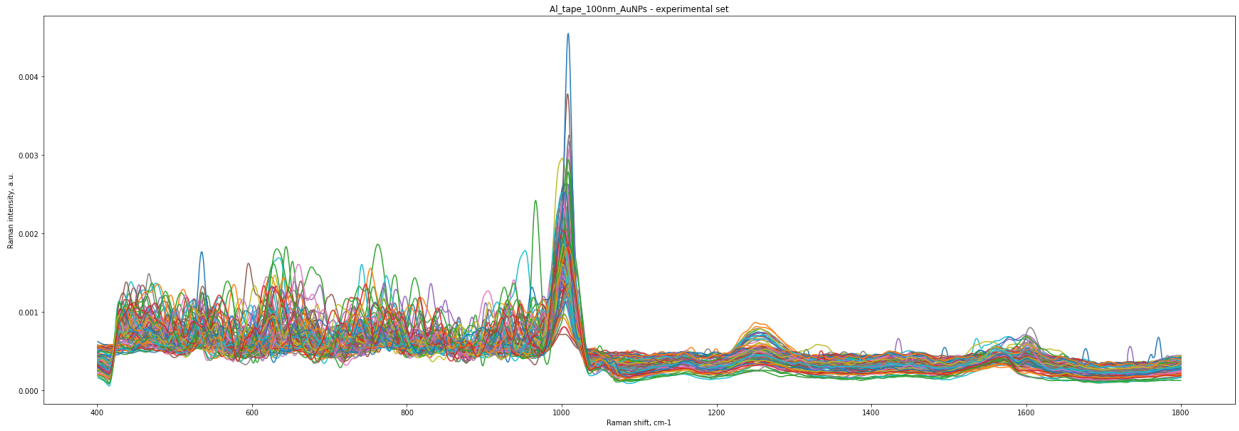
Create plots for experimental sets

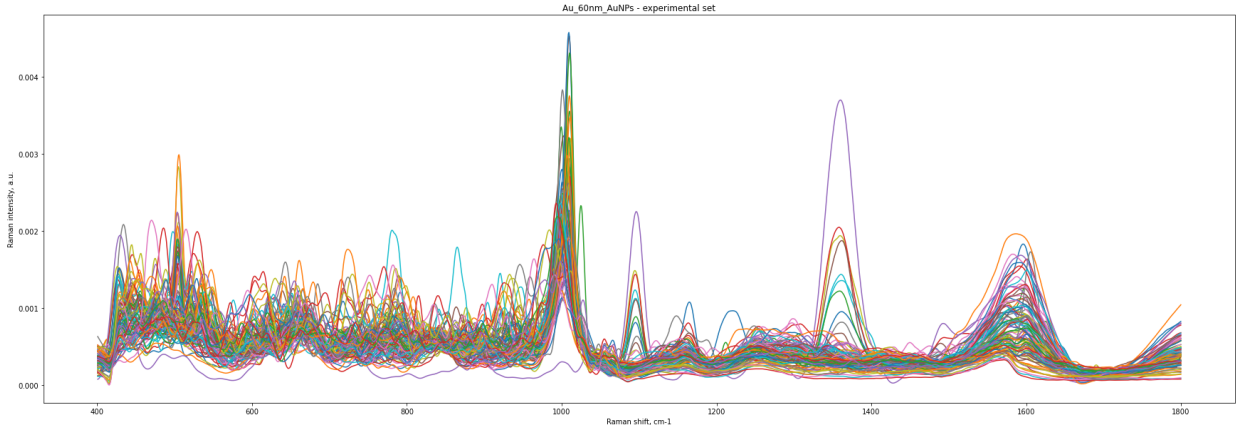
In [8]:
```
# Combine every plot for each patient
# plots every spectra for one experimental set as defined in exp_set
for key_set in processed_urine_spectra.keys():
    if key_set != 'Si_60nm_AuNPs':
        plt.figure(figsize =(30,10))
        plt.xlabel('Raman shift, cm-1')
        plt.ylabel('Raman intensity, a.u.')
        plt.title(f'{key_set} - experimental set')
        for key_ID in processed_urine_spectra[key_set].keys():
            plt.plot(wave[0], processed_urine_spectra[key_set][key_ID])
        len(processed_urine_spectra[key_set])
```

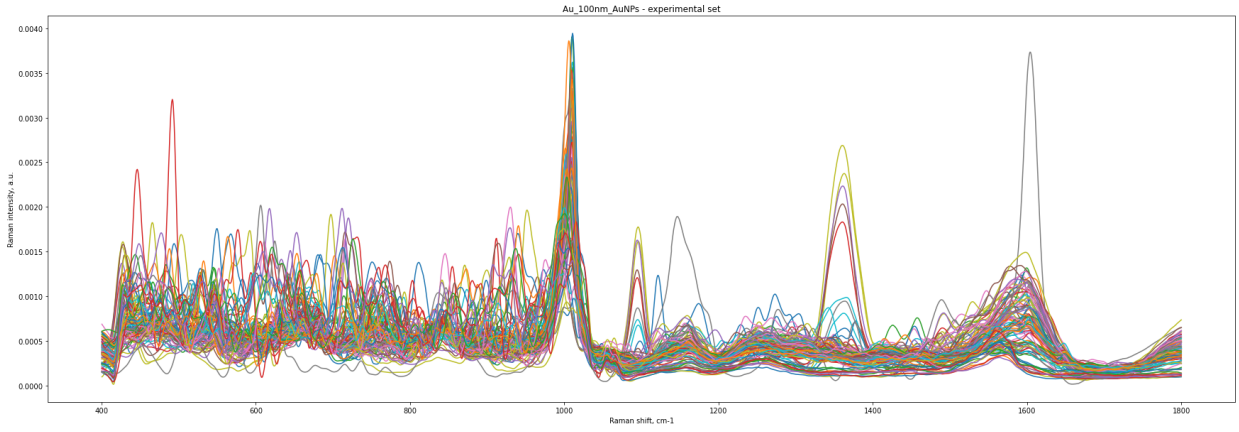












Save processed spectra

In [9]:
```
# save output dictionaries into pickle files
```

```
import pickle
with open("processed_urine_spectra.pkl","wb") as file:
    pickle.dump(processed_urine_spectra,file)
```