```
%{
This function takes in the extracted time-domain features and
 organises the data:
1. For each single dataset (each body part) we concatenate
 horizontally
    all the features after removing the columns (features) that are
 not
    needed eg magnetometere
2. We then concatetane vertically all training samples
    -> result in having a single matrix per activity that will have
 all
    training samples of all features + parts of body
3. We delete the columns containing magnetometer data as they are not
relevant to the purpose of the classifiers we are building

Arguments:
- `processedData`  -> the struct containing the extracted and reduced
time-domain features

Returns:
- `arrayPerActivity` -> struct containing a single array per
activity without any magnetometer data.
%}

function arrayPerActivity = organiseFeatures(processedData)

    % array containing the names of the activities.
    % These names will match the field names in the struct
    sets = ["LGW","RA","RD","SiS","StS"];
    % array containing the names of the time domain features.
    % These names will match the field names in the struct.
    features = ["MAX", "MIN", "AVG", "SD","RMS", "ZC", "MSC"];
    % define the IMU data to include in the data
    included_readings = ["gyro", "accel", "magnet"];

    % ----------------------------------------------
    % 1. Loop through the processed data and horizonatally concatenate
    % all the features
    % ----------------------------------------------
    for ff = 1 : length(sets)
        for kk = 1 : length(processedData)
            % sample is the struct containing the 7 tables
            % (one for each time feature)
            sample = processedData(kk).(sets{ff});
            % extract each of the 7 features
            sample_max = sample(1).MAX;
            sample_min = sample(1).MIN;
            sample_avg = sample(1).AVG;
            sample_sd = sample(1).SD;
            sample_rms = sample(1).RMS;
            sample_zc = sample(1).ZC;
            sample_msc = sample(1).MSC;
```

```
           % concatenate horizontally all of the features for this
dataset
           sample_feature_collection = [sample_max, sample_min,
sample_avg, sample_sd, sample_rms, sample_zc, sample_msc];
           features_collected(kk).(sets{ff}) =
sample_feature_collection;
       end
   end

   % ---------------------------------------------
   % 2. Loop through the features and vertically all the training
samples for
   % each activity. Result in a single table for each activity
   % ---------------------------------------------
   % loop through each of the folders
   for ff = 1 : length(sets)
       for kk = 1 : length(features_collected)
           % sample is the table containing the concatenated features
as
           % a single table
           sample = features_collected(kk).(sets{ff});
           if kk == 1
               growing_activity_data = sample;
           else
               % vertically concatenate all the activity's data
               growing_activity_data =
vertcat(growing_activity_data,sample);
           end
       end
       % array_per_activity is a struct that will contain a table for
each
       % activity.
       array_per_activity(1).(sets{ff}) = growing_activity_data;
       % reset the variable for the next activity
       clearvars growing_activity_data
   end


   % ---------------------------------------------
   % 3. Remove the magnetometer columns from the data
   % ---------------------------------------------
   fprintf("\nExcluding Magnetometer data...\n")
   for ff = 1 : length(sets)
       sample = array_per_activity(1).(sets{ff});
       % DUPLICATE SAMPLE
       update_sample = sample;
       sample_dim = size(update_sample);
       for ii=6 : 6 : sample_dim(2)
           % if deleting magnetometer data, find every sixth column
and delete the
           % next 3 after it.
           % fprintf("\ncolumn %i of %i\n", ii, sample_dim(2));
           update_sample(:,ii+1) = [];
           update_sample(:,ii+1) = [];
```

```
            update_sample(:,ii+1) = [];
            % update the width because the array shrinks as we go
 along
            sample_dim = size(update_sample);
            % manually exit the loop when we reach the end
            if ii == sample_dim(2)
                break
            end
        end
        % update the table for each activity with the new one without
 the
        % magnetometer data
        arrayPerActivity(1).(sets{ff}) = update_sample;
    end


    %{
    Each activity now has one array in arrayPerActivity.
    Each of those array has a size of Nx294
        63 columns from the raw data after removing the timestamps X
        7 extracted time domain features X
        (2/3) getting rid of magnetometer readings (deleting 3 in
 every 9 columns)
        so 63 x 7 x (2/3) = 294

    The N varies with each dataset depending on the original number of
 rows (samples) in
    the raw data.
    %}

end
```

Published with MATLAB® R2020a