```
import os
import matplotlib.pyplot as plt
import descarteslabs as dl
import numpy as np
from sys import exit
import sklearn
from sklearn import svm
from sklearn.preprocessing import StandardScaler
from celery import Celery

####################
# Function         #
####################
### Running mean/Moving average
def rolling_median(var,window):
    '''var: array-like. One dimension
    window: Must be odd'''
    n=len(var)
    halfW=int(window/2)
    med=np.zeros(shape=(var.shape))
    for j in range(halfW,n-halfW):
        med[j]=np.ma.median(var[j-halfW:j+halfW+1])

    for j in range(0,halfW):
        w=2*j+1
        med[j]=np.ma.median(var[j-w/2:j+w/2+1])
        i=n-j-1
        med[i]=np.ma.median(var[i-w/2:i+w/2+1])

    return med
####################

# Run in parallel on Google Cloud
celery = Celery('compute_ndvi_forCloud', broker='redis://localhost:6379/0')

vlen=992
hlen=992
start='2001-01-01'
end='2016-12-31'
nyears=16
country='Brazil'
makePlots=False
padding = 16
pixels = vlen+2*padding
res = 120.0

matches=dl.places.find('united-states_illinois')
aoi = matches[0]
shape = dl.places.shape(aoi['slug'], geom='low')

dltiles = dl.raster.dltiles_from_shape(res, vlen, padding, shape)
lonlist=np.zeros(shape=(len(dltiles['features'])))
latlist=np.zeros(shape=(len(dltiles['features'])))
```

```
for i in range(len(dltiles['features'])):
    lonlist[i]=dltiles['features'][i]['geometry']['coordinates'][0][0][0]
    latlist[i]=dltiles['features'][i]['geometry']['coordinates'][0][0][1]
#
features=np.zeros(shape=(len(dltiles['features']),nyears,pixels*pixels,6))
target=np.zeros(shape=(len(dltiles['features']),nyears,pixels*pixels))

@celery.task
def tile_function(dltile,makePlots=False):

    clas=["" for x in range(7)]
    clasLong=["" for x in range(255)]
    clasDict={}
    clasNumDict={}
    f=open(wd+'data/ground_data.txt')
    for line in f:
        tmp=line.split(',')
        clasNumLong=int(tmp[0])
        clasLong[clasNumLong]=tmp[1]
        clasNum=int(tmp[3])
        clas[clasNum]=tmp[2]

        clasDict[clasLong[clasNumLong]]=clas[clasNum]
        clasNumDict[clasNumLong]=clasNum


    lon=dltile['geometry']['coordinates'][0][0][0]
    lat=dltile['geometry']['coordinates'][0][0][1]
    globals().update(locals())
    print lon
    print lat
    latsave=str(lat)
    latsave=latsave.replace('.','-')
    lonsave=str(lon)
    lonsave=lonsave.replace('.','-')

#    print '\n\n'
#    print 'dltile: '+str(tile)+' of '+str(len(dltiles['features']))

    ###############################################
    # Find Ground Classification data
    ###############################################

    images = dl.metadata.search(
        const_id=["CDL","CDL"],
        geom=dltile['geometry'],
        limit = 2000
        )

    n_images = len(images['features'])
    #    print('Number of image matches: %d' % n_images)

    year=np.zeros(shape=(n_images),dtype='int')
    j=-1
```

```
for feature in images['features']:
    j+=1
    scene=feature['id']

    year[j]=int(scene[14:18])
    if j==0:
        maxyear=year[j]
        maxj=j
        maxscene=scene
        continue

    if year[j]>maxyear:
        maxyear=year[j]
        maxj=j
        maxscene=scene


cdl = dl.raster.get_bands_by_constellation("CDL").keys()
cdl1 = dl.raster.get_bands_by_constellation("CDL").keys()
avail_bands = set(cdl).intersection(cdl1)
#    print('Available bands: %s' % ', '.join([a for a in avail_bands]))

band_info = dl.raster.get_bands_by_constellation("CDL")

globals().update(locals())
try:
    valid_range = band_info['class']['valid_range']
    arr, meta = dl.raster.ndarray(
        maxscene,
        resolution=dltile['properties']['resolution'],
        bounds=dltile['properties']['outputBounds'],
        srs=dltile['properties']['cs_code'],
        bands=['class'],
        scales=[[valid_range[0], valid_range[1]]],
        data_type='Float32'
        )
except:
    print('class: %s could not be retreived' % maxscene)

arr=arr.astype(int)

arrClas=np.zeros(shape=(arr.shape))
for v in range(pixels):
    for h in range(pixels):
        arrClas[v,h]=clasNumDict[arr[v,h]]

if makePlots:
    if not os.path.exists(r'../figures/'+country+'/'+str(lon)+'_'+str(lat)):
        os.makedirs(r'../figures/'+country+'/'+str(lon)+'_'+str(lat))
    plt.figure(figsize=[16,16])
    plt.imshow(arrClas, cmap='jet', vmin=0, vmax=11)
    #plt.title('NDVI: '+str(lon)+'_'+str(lat)+', '+str(date), fontsize=20)
    plt.colorbar()
    #cb.set_label("Cloud")
    plt.savefig(wd+'figures/'+country+'/'+str(lon)+'_'+str(lat)+'/groud_data_simpl
    plt.clf()

if np.sum(arrClas)==0:
    print 'No Data: In the Ocean'
    return

oceanMask=np.zeros(shape=(arrClas.shape),dtype=bool)
for v in range(pixels):
    for h in range(pixels):
        if arrClas[v,h]==0:
            oceanMask[v,h]=True
###############################################

images = dl.metadata.search(
    const_id=["MO", "MY"],
    start_time=start,
    end_time=end,
    geom=dltile['geometry'],
    cloud_fraction=0.8,
    limit = 2000
    )

n_images = len(images['features'])
print('Number of image matches: %d' % n_images)
mo = dl.raster.get_bands_by_constellation("MO").keys()
my = dl.raster.get_bands_by_constellation("MY").keys()
avail_bands = set(mo).intersection(my)
print('Available bands: %s' % ', '.join([a for a in avail_bands]))

band_info = dl.raster.get_bands_by_constellation("MO")

dayOfYear=np.zeros(shape=(n_images))
year=np.zeros(shape=(n_images),dtype=int)
month=np.zeros(shape=(n_images),dtype=int)
day=np.zeros(shape=(n_images),dtype=int)
plotYear=np.zeros(shape=(n_images))
xtime=[]
i=-1
for feature in images['features']:
    i+=1
    # get the scene id
    scene = feature['id']

    xtime.append(str(images['features'][i]['id'][20:30]))
    date=xtime[i]
    year[i]=xtime[i][0:4]
    month[i]=xtime[i][5:7]
    day[i]=xtime[i][8:10]
    dayOfYear[i]=(float(month[i])-1)*30+float(day[i])
    plotYear[i]=year[i]+dayOfYear[i]/365.0


indexSorted=np.argsort(plotYear)
```

```
    ####################
    # Define Variables #
    ####################
    ndviAll=-9999*np.ones(shape=(pixels,pixels,n_images))
    ndwiAll=np.zeros(shape=(pixels,pixels,n_images))
#    cloudAll=-9999*np.ones(shape=(pixels,pixels,n_images))
    Mask=np.ones(shape=(pixels,pixels,n_images),dtype=bool)
    dayOfYear=np.zeros(shape=(n_images))
    year=np.zeros(shape=(n_images))
    month=np.zeros(shape=(n_images))
    day=np.zeros(shape=(n_images))
    plotYear=np.zeros(shape=(n_images))
#    ndviHist=np.zeros(shape=(40,n_images))
#    ndviAvg=np.zeros(shape=(n_images))
#    ndviMed=np.zeros(shape=(n_images))
    xtime=[]

    ndviHist=np.zeros(shape=(40,n_images))
    ndviAvg=np.zeros(shape=(n_images))
    ndviMed=np.zeros(shape=(n_images))
    ####################
    k=-1

    for j in range(len(indexSorted)):
#    for j in range(10):
        # get the scene id
        scene = images['features'][indexSorted[j]]['key']
        ###############################################
        # NDVI
        ###############################################
        # load the image data into a numpy array
        try:
            valid_range = band_info['ndvi']['valid_range']
            physical_range = band_info['ndvi']['physical_range']
            arrNDVI, meta = dl.raster.ndarray(
                scene,
                resolution=dltile['properties']['resolution'],
                bounds=dltile['properties']['outputBounds'],
                srs=dltile['properties']['cs_code'],
                bands=['ndvi', 'alpha'],
                scales=[[valid_range[0], valid_range[1], physical_range[0], physical_r
                data_type='Float32'
                )
        except:
            print('ndvi: %s could not be retreived' % scene)
            continue

        ###############################################
        # Test for bad days
        ###############################################

        #take out days without data
        if arrNDVI.shape == ()==True:
```

```
        continue
    maskforNDVI = arrNDVI[:, :, 1] != 0 # False=Good, True=Bad
    if np.sum(maskforNDVI)==0:
        print 'continued'
        continue

    #######################
    # Get cloud data      #
    #######################
    try:
        valid_range = band_info['visual_cloud_mask']['valid_range']
        physical_range = band_info['visual_cloud_mask']['physical_range']
        arrCloud, meta = dl.raster.ndarray(
            scene,
            resolution=dltile['properties']['resolution'],
            bounds=dltile['properties']['outputBounds'],
            srs=dltile['properties']['cs_code'],
            bands=['visual_cloud_mask', 'alpha'],
            scales=[[valid_range[0], valid_range[1]]],
            data_type='Float32'
            )
    except:
        print('cloud: %s could not be retreived' % scene)
        continue
    #######################

    #### Only for Desert ####
#    for v in range(pixels):
#        for h in range(pixels):
#            arrCloud[:,:,0]=0
    #### Only for Desert ####

    # take out days with too many clouds
    maskforCloud = arrCloud[:, :, 0] == 0
    if np.sum(maskforCloud)<0.1*(pixels*pixels):
        print 'clouds: continued'
        continue
    k+=1

    ###############################################
    # time
    ###############################################

    xtime.append(str(images['features'][indexSorted[j]]['id'][20:30]))
    date=xtime[k]
    year[k]=xtime[k][0:4]
    month[k]=xtime[k][5:7]
    day[k]=xtime[k][8:10]
    dayOfYear[k]=(float(month[k])-1)*30+float(day[k])
    plotYear[k]=year[k]+dayOfYear[k]/365.0

    ###############################################
    # Back to NDVI
    ###############################################
```

```
        print date, k
        sys.stdout.flush()
        maskforCloud = arrCloud[:, :, 0] != 0
        #maskforCloud = arrCloud[:, :, 1] == 0 #for desert
        maskforNDVI = arrNDVI[:, :, 1] == 0

        for v in range(pixels):
            for h in range(pixels):
                if maskforCloud[v,h]==0 and maskforNDVI[v,h]==0 and oceanMask[v,h]==0:
                    Mask[v,h,k]=0

        if makePlots:

            if not os.path.exists(r'../figures/'+country+'/'+str(lon)+'_'+str(lat)):
                os.makedirs(r'../figures/'+country+'/'+str(lon)+'_'+str(lat))

            masked_ndvi = np.ma.masked_array(arrNDVI[:, :, 0], Mask[:,:,k])
            plt.figure(figsize=[16,16])
            plt.imshow(masked_ndvi, cmap='jet', vmin=-1, vmax=1)
            plt.title('NDVI: '+str(lon)+'_'+str(lat)+', '+str(date), fontsize=20)
            cb = plt.colorbar()
            cb.set_label("NDVI")
            plt.savefig(wd+'figures/'+country+'/'+str(lon)+'_'+str(lat)+'/ndvi_'+str(d
            plt.clf()

        ndviAll[:,:,k]=np.ma.masked_array(arrNDVI[:,:,0],Mask[:,:,k])

        ###############################################
        # Cloud
        ###############################################

        if makePlots:
            #masked_cloud = np.ma.masked_array(arrCloud[:, :, 0], maskforCloud)
            masked_cloud = arrCloud[:, :, 0]
            plt.figure(figsize=[16,16])
            plt.imshow(masked_cloud, cmap='gray', vmin=0, vmax=1)
            plt.title('Cloud: '+str(lon)+'_'+str(lat)+', '+str(date), fontsize=20)
            cb = plt.colorbar()
            cb.set_label("Cloud")
            plt.savefig(wd+'figures/'+country+'/'+str(lon)+'_'+str(lat)+'/cloud_'+str(
            plt.clf()

#        for v in range(pixels):
#            for h in range(pixels):
#                cloudAll[v,h,k]=arrCloud[v,h,0]


        ###############################################
        # NDWI
        ###############################################

        try:
            valid_range = band_info['nir']['valid_range']
```

```
    physical_range = band_info['nir']['physical_range']
    nir, meta = dl.raster.ndarray(
        scene,
        resolution=dltile['properties']['resolution'],
        bounds=dltile['properties']['outputBounds'],
        srs=dltile['properties']['cs_code'],
        bands=['nir', 'alpha'],
        scales=[[valid_range[0], valid_range[1], physical_range[0], physical_r
        data_type='Float32'
        )
except:
    print('nir: %s could not be retreived' % scene)
    continue

nirM=np.ma.masked_array(nir[:,:,0],Mask[:,:,k])

try:
    valid_range = band_info['green']['valid_range']
    physical_range = band_info['green']['physical_range']
    green, meta = dl.raster.ndarray(
        scene,
        resolution=dltile['properties']['resolution'],
        bounds=dltile['properties']['outputBounds'],
        srs=dltile['properties']['cs_code'],
        bands=['green', 'alpha'],
        scales=[[valid_range[0], valid_range[1], physical_range[0], physical_r
        data_type='Float32'
        )
except:
    print('green: %s could not be retreived' % scene)
    continue

greenM=np.ma.masked_array(green[:,:,0],Mask[:,:,k])

for v in range(pixels):
    for h in range(pixels):
        if oceanMask[v,h]==True:
            continue
        ndwiAll[v,h,k] = (greenM[v,h]-nirM[v,h])/(nirM[v,h]+greenM[v,h]+1e-9)
#                ndwiAll[v,h,k] = np.clip(128.*(ndwiAll[v,h,k]+1), 0, 255).ast

if makePlots:
    #masked_cloud = np.ma.masked_array(arrCloud[:, :, 0], maskforCloud)
    masked_cloud = ndwiAll[:, :, k]
    plt.figure(figsize=[16,16])
    plt.imshow(ndwiAll[:,:,k], cmap='jet', vmin=-1, vmax=1)
    plt.title('NDWI: '+str(lon)+'_'+str(lat)+', '+str(date), fontsize=20)
    cb = plt.colorbar()
    cb.set_label("NDWI")
    plt.savefig(wd+'figures/'+country+'/'+str(lon)+'_'+str(lat)+'/ndwi_'+str(d
    plt.clf()

globals().update(locals())
```

```
        ############################
        # Variables for Histogram  #
        ############################
        ndviRavel=arrNDVI[:,:,0].ravel()
        #        cloudRavel=arrCloud[:,:,0].ravel()
        MaskRavel=Mask[:,:,0].ravel()

        #        cloud_mask=cloudRavel != 0
        ndviWithMask=np.ma.masked_array(ndviRavel, MaskRavel)
        hist,edges=np.histogram(ndviWithMask,bins=np.arange(-1.,1.01,.05))
        ndviHist[:,k]=hist
        ndviAvg[k]=np.mean(ndviWithMask)
        ndviMed[k]=np.median(ndviWithMask)
        ############################

    #return ndviAll,cloudAll,ndviHist,ndviAvg,plotYear,k


#    if makePlots:
#        plt.clf()
#        for v in range(pixels):
#            for h in range(pixels):
#                plt.figure(1)
#                ndviWithMask = np.ma.masked_array(ndviAll[v,h], Mask[v,h])
#                plt.plot(plotYear,ndviWithMask,'.', color=(float(h)/float(pixels), 0.
#                plt.ylim([-1.,1,])
#                plt.xlabel('year')
#                plt.ylabel('ndvi')
#                plt.title('ndvi 2016 pixel '+str(v)+'_'+str(h))
#                plt.savefig(wd+'figures/'+country+'/'+str(lon)+'_'+str(lat)+'/cloud_m
#                plt.clf()

    plt.clf()
    plotYeartwoD=np.zeros(shape=(40,k))
    yvalue=np.zeros(shape=(40,k))
    for d in range(k):
        for v in range(40):
            plotYeartwoD[v,d]=plotYear[d]
            yvalue[v,d]=edges[v]



#        rollingmed=rolling_median(ndviAvg[0:k],10)
#    rollingmed=rolling_median(ndviMed[0:k],10)

    x2=plotYear[0:k]
#    ydata2=ndviAvg[0:k]
    ydata2=ndviMed[0:k]
    yfit2=movingaverage(ydata2,16)

    plt.clf()
    plt.figure(1)
    plt.contourf(plotYeartwoD[:,0:k],yvalue[:,0:k],ndviHist[:,0:k],100,cmap=plt.cm.gis
    plt.colorbar()
    plt.plot(x2[2:k-2],yfit2[2:k-2],'.k',linewidth=1)
    #plt.plot(plotYeartwoD[0,0:k],ndviAvg[0:k],'*',color='k')
    plt.title('NDVI 2016 '+str(lon)+'_'+str(lat))
    plt.xlabel('date')
    plt.ylabel('ndvi')
    plt.ylim(-1,1)
    plt.savefig(wd+'figures/'+country+'/'+str(lon)+'_'+str(lat)+'/_heatmap.pdf')
    plt.clf()

#        plt.plot(x2[2:k-2],yfit2[2:k-2],'.k',linewidth=1)
#        plt.title('NDVI 2016 '+str(lon)+'_'+str(lat))
#        plt.ylim(-1,1)
#        plt.xlabel('date')
#        plt.ylabel('ndvi')
#        plt.savefig(wd+'test_fig'+'_avgline.pdf')
#        plt.clf()
#
#        plt.plot(rollingmed[2:])
#        plt.ylim(-1,1)
#        plt.savefig(wd+'test_fig'+'_rolling_max.pdf')
    globals().update(locals())

    ########################
    # Save variables       #
    ########################
    print lat,lon

    if not os.path.exists(r'../saved_vars/'+str(lon)+'_'+str(lat)):
        os.makedirs(r'../saved_vars/'+str(lon)+'_'+str(lat))

    np.save(wd+'saved_vars/'+str(lon)+'_'+str(lat)+'/ndwiAll',ndwiAll)
    np.save(wd+'saved_vars/'+str(lon)+'_'+str(lat)+'/Mask',Mask)
    np.save(wd+'saved_vars/'+str(lon)+'_'+str(lat)+'/oceanMask',oceanMask)
    np.save(wd+'saved_vars/'+str(lon)+'_'+str(lat)+'/plotYear',plotYear)
##    np.save(wd+'saved_vars/'+country+'/'+str(lon)+'_'+str(lat)+'/ndviHist',ndviHist)
##    np.save(wd+'saved_vars/'+country+'/'+str(lon)+'_'+str(lat)+'/ndviAvg',ndviAvg)
##    np.save(wd+'saved_vars/'+country+'/'+str(lon)+'_'+str(lat)+'/ndviAvg',ndviMed)
    np.save(wd+'saved_vars/'+str(lon)+'_'+str(lat)+'/n_good_days',int(k))
    np.save(wd+'saved_vars/'+str(lon)+'_'+str(lat)+'/month',month)
    np.save(wd+'saved_vars/'+str(lon)+'_'+str(lat)+'/year',year)
##    np.save(wd+'saved_vars/'+country+'/'+str(lon)+'_'+str(lat)+'/edges',edges)
    np.save(wd+'saved_vars/'+str(lon)+'_'+str(lat)+'/arrClas',arrClas)
    np.save(wd+'saved_vars/'+str(lon)+'_'+str(lat)+'/ndviAll',ndviAll)

    ### Upload files to bucket
    ### Delete files

for tile in range(len(dltiles['features'])):
    tile=29
    dltile=dltiles['features'][tile]
    tile_function(dltile)


#for i in range(len(dltiles['features'])):
```

```
#    ## Check in the bucket
#    ## gsutil ls
#    if not os.path.exists(r'../saved_vars/'+str(lonlist[i])+'_'+str(latlist[i])+'/ndv
#        dltile=dltiles['features'][i]
#        tile_function(dltile)
#
```