```python
# -*- coding: utf-8 -*-

import os
from os import listdir
from os.path import isfile, join
import pathlib
import gzip

DEBUG = False


def humanbytes(B):
    """Return the given bytes as a human friendly KB, MB, GB, or TB string."""
    B = float(B)
    KB = float(1024)
    MB = float(KB ** 2)  # 1,048,576
    GB = float(KB ** 3)  # 1,073,741,824
    TB = float(KB ** 4)  # 1,099,511,627,776

    if B < KB:
        return '{0} {1}'.format(B, 'Bytes' if 0 == B > 1 else 'Byte')
    elif KB <= B < MB:
        return '{0:.2f} KB'.format(B / KB)
    elif MB <= B < GB:
        return '{0:.2f} MB'.format(B / MB)
    elif GB <= B < TB:
        return '{0:.2f} GB'.format(B / GB)
    elif TB <= B:
        return '{0:.2f} TB'.format(B / TB)


def getFileTypeInfo(x):
    lastSuffix = pathlib.Path(x).suffix
    lastSuffix2 = pathlib.Path(x.rstrip(lastSuffix)).suffix

    lastSuffix = lastSuffix.lstrip('.')
    lastSuffix2 = lastSuffix2.lstrip('.')

    if DEBUG:
        print('getFileTypeInfo -> inputValue: ' + x)
        print('getFileTypeInfo -> lastSuffix: ' + lastSuffix)
        print('getFileTypeInfo -> lastSuffix2:' + lastSuffix2)

    retStr = ''

    if eligibleforcompression(x):
        retStr = '[*]'
    else:
        retStr = '---'

    if len(lastSuffix) > 0:
        retStr = retStr + "[" + lastSuffix.rjust(5, ' ') + "]"
    else:
        retStr = retStr + '[   ]'

    if len(lastSuffix2) > 0:
        retStr = retStr + "[" + lastSuffix2.rjust(5, ' ') + "]"
    else:
        retStr = retStr + '-------'

    return retStr


def eligibleforcompression(x):
    eligExtensions = ['.csv', '.pkl']
    eligible = False

    for ext in eligExtensions:
        if DEBUG:
            print("eligibleforcompress -> eligible extension:    " + ext)
            print("eligibleforcompress -> extension received: " + pathlib.Path(x).suffix)
        if ext == pathlib.Path(x).suffix:
            eligible = True
    if DEBUG:
        print(f'{x} is eligible for compression. Return with {str(eligible)}')
    return eligible


def compressfile(x, abspath, removeOriginal=False):
    if DEBUG:
        print(f'compressfile has been called on: {x}')

    if eligibleforcompression(x):
        if DEBUG:
            print(x + " is eligible for compression")
    else:
        if DEBUG:
            print(x + " is not eligible for compression")
        return 0


    file = open(join(abspath, x), "rb")
    if DEBUG:
        print(x + " file has been opened")
    data = file.read()
    bindata = bytearray(data)
    print("======> compressing file: " + x)
    compressName = join(abspath, x) + ".gz"
    with gzip.open(compressName, "wb") as f:
        f.write(bindata)

    if removeOriginal:
        if DEBUG:
            print(f'Compressed file. Removing original: {x}')
        os.remove(join(abspath, x))

    return 1


def exploreDirectory(absPath, compress=False, removeOriginal=False):
    onlyFiles = [f for f in listdir(absPath) if isfile(join(absPath, f))]
    onlyDirs = [f for f in listdir(absPath) if not isfile(join(absPath, f))]
    onlyFiles.sort()
    onlyDirs.sort()
    numCompressedFiles = 0

    if DEBUG:
        print(f'exploreDirectory called with directory:       {str(absPath)}')
        print(f'exploreDirectory called with compress:        {str(compress)}')
        print(f'exploreDirectory called with removeOriginal:  {str(removeOriginal)}')

    if compress:
        print("Scanning [" + absPath + "] for data files to compress")
        if len(onlyFiles) > 0:
            print(f"======> {len(onlyFiles)} files found in directory.")
            for x in onlyFiles:
                numCompressedFiles += compressfile(x, absPath, removeOriginal=removeOriginal)
            print(f"======> {numCompressedFiles} file(s) have been compressed.")
        else:
            print("======> No files found in directory
.")

        print('')
        for x in onlyDirs:
            exploreDirectory(join(absPath,x), compress=True, removeOriginal=removeOriginal)


        if DEBUG and compress:
            print("Explore directory completed compression.")


    else:

        if len(onlyFiles) == 0:
            print(f'[D] {absPath} [Empty directory]')
        else:
            print(f'[D] {absPath}')
            for x in onlyFiles:
                print(f'{getFileTypeInfo(x)}--> {x} ({humanbytes(os.stat(join(absPath, x)).st_size)})')

        print('')
        for x in onlyDirs:
            exploreDirectory(join(absPath, x))

```