松 山 大 学 論 集
第30巻第４・２号抜刷
２０１８ 年 10 月 発 行

クローリング EDINET

中　溝　晃　介

クローリング EDINET

中　溝　晃　介

第 1 節　は　じ　め　に

近年では，会計研究において用いるデータが多種多様になってきている。個人パソコンの性能が上がり，データ分析の方法にも複数の選択肢がみられるようになった。特に，機械学習ひいては人工知能の分野は会計分野に限らず注目を集めており，情報分野を超えてあらゆる分析に適用されている。これらの分析にはデータセットが不可欠であり，データ収集を行ってから前処理と呼ばれるデータを整形することが求められる。分析の結果はこの前処理にかかっており，労力の大半が前処理にかかるとも言われている。

会計分野では，前処理の前段階であるデータ収集が困難であることが問題である。例えば，機械処理に代表される画像認識では，実験用に 1,000 万を超えるデータがインターネットを通して無償で提供されている。画像認識に取り組む人口が多いことも味方しているだろう。会計分野のデータは日経 NEEDS のような有料のデータベースに頼っているところが現状である。金融庁が運営する EDINET や東京証券取引所が運営する TDnet では，企業の会計報告書が無償で提供されているが，１年間の上場企業データを一括でダウンロードする，といった処理ができない。手作業にも限界があるので，多く利用者がデータを取りにくいと感じていることだろう。そこで，本稿では自動で会計報告書をダウンロードするプログラムを作成することを試みる。

本稿の目的は，研究に用いるデータセットを作成することである。具体的には，金融庁の EDINET から XBRL 形式の有価証券報告書や四半期報告書，半

期報告書をダウンロードし，必要なデータを抽出する作業を自動化させる。日経NEEDSなどのデータベースと同じデータもあれば，入手することが難しいデータも対象となる。

本稿では，クローリングとスクレイピングの技術を使用している。クローラーを使ってデータを収集することをクローリング（Crawling）と呼び，Webページから必要な情報を抜き出すことをスクレイピング（Scraping）と呼ぶ[1)]。

なお，本稿ではプログラミング言語Pythonを使用している。Pythonおよび各種ライブラリのバージョンについては本稿末尾に記載している。

第2節 EDINETコードの入手

金融商品取引法に基づくディスクロージャー制度は，有価証券報告書等開示書類の提出，財務局による受理，審査および縦覧という一連の手続からなっている。この一連の手続は，従前は紙媒体で行われていたが，2004年6月から通信回線を利用した電子開示システムEDINETを通して行うことが義務付けられるようになった。EDINETによる開示は，2001年6月より順次任意で実施され，2004年6月からは，任意電子開示手続である有価証券通知書・発行登録通知書等を除き，義務化されている。2001年6月の電子化当初のEDINETによる開示書類の提出会社数（内国会社）は，約500社であったが，2006年6月末には約5,100社となっている[2)]。

日本のEDINETには難点がいくつか存在していた。第一に，データの再利用が難しいことが挙げられる。第二に多言語対応が基本的に考えられていないことである。このような問題に対応するため，金融庁では2004年11月24日にEDINETの高度化に関する協議会が発足し，機能の充実をはかるための検討を始めた。2005年5月には「有価証券報告書等に関する業務の業務・システム見直し方針案」が公表され，このなかでEDINETの機能を充実させるために

1）加藤［2017］，3頁。
2）小谷［2007］，43頁（筆者一部修正）。

「XBRL 化に向けた動きを加速する」ことが表明されたのである。その後，2008 年 4 月より始まる会計年度より，企業が提出する有価証券報告書のうち財務諸表部分については，XBRL 形式での提出が義務付けられたのである[3]。なお，2013 年 4 月より始まる会計年度より，XBRL 化の対象は有価証券報告書全体へと拡張された。

財務諸表の分析という観点に立てば，対象となる書類は内国法人が提出する有価証券報告書，半期報告書，四半期報告書が中心となる。EDINET を通して報告書を入手する場合，EDINET コードと呼ばれる，法人または個人等の開示書類等提出者毎に付番される一意のコードを利用することになる。企業名でも検索することは可能であるが，企業名を使う場合，必ずしも特定の企業のみが結果として表れるわけではないため，EDINET コードを使用する方が良い。

EDINET コードのリストは EDINET のホームページ[4] の「ダウンロード」のページから入手することができる。EDINET コードリストには，EDINET コード，提出者種別，上場区分，連結の有無，資本金，決算日，提出者名，提出者名（英字），提出者名（ヨミ），所在地，提出者業種，証券コード，提出者法人番号がデータとして含まれている。

EDINET に提出を行う者の種類は，①内国法人・組合，②外国法人・組合，③外国政府等，④個人（組合発行者を除く），⑤個人（非居住者）（組合発行者を除く），⑥内国法人・組合（有価証券報告書等の提出義務者以外），⑦外国法人・組合（有価証券報告書等の提出義務者以外）の 7 種である。このうち，有価証券報告書を提出する者は①内国法人・組合と②外国法人・組合の 2 つである[5]。

3）坂上［2011］，24 頁。
4）http://disclosure.edinet-fsa.go.jp/
5）金融庁［2018］，39-40，47 頁。
⑥内国法人・組合（有価証券報告書等の提出義務者以外）の企業でも EDINET 上で有価証券報告書の提出を確認することができる。これは，過去に提出対象となっていた企業等の理由が考えられる。

ここで，例としてEDINETコードリストから作成した2018年2月8日時点と2018年7月8日時点の提出者数を【表1】に示す。

表1 EDINETにおける提出者数の比較

提出者	2018年2月8日	2018年7月8日
内国法人・組合	4,419	4,412
外国法人・組合	191	183
外国政府等	56	53
個人（組合発行者を除く）	2,729	2,716
個人（非居住者）（組合発行者を除く）	76	77
内国法人・組合（有価証券報告書等の提出義務者以外）	1,684	1,649
外国法人・組合（有価証券報告書等の提出義務者以外）	643	635
合計	9,798	9,725

【表1】を見ると，5ヶ月の間で提出者数に差異が生じていることがわかる。このEDINETコードリストは日毎に更新されるため，上場廃止となった企業などリストから外れる企業が存在し，同じ提出者数とはならないことに注意しなければならない。

別の方法として，証券コードを利用することもできる。東京証券取引所のホームページでは「東証上場銘柄一覧」の中に証券コードが月末毎に公開されており，証券コードと対応するEDINETコードリストを抽出し利用する。この場合，上場企業に限定されてしまうが，日によって企業数が異なるという問題は生じなくなる。本稿では両方のコードを掲載することにする。EDINETコードリストのみからEDINETを取得する方法を【リスト1】に示し，東京証券取引所の証券コードとEDINETコードリストからEDINETコードを取得する方法を【リスト2】に示す。

【リスト1】では，ダウンロードしたファイル“EdinetcodeDlInfo.csv”をpandasで読み込んでいる。その際，ファイルに含まれる日本語表記の文字列が文字化けすることを防ぐために，JISコードなどの条件を付している。1行目と2行

目は不要かつ 1 行目が列名になることを防ぐために，列名を指定して読み込んでいる。

リスト 1

```
import pandas as pd
import codecs

with codecs.open('EdinetcodeDlInfo.csv', "r", "Shift-JIS", "ignore"
) as file:
df1 = pd.read_csv(file, names= ('ＥＤＩＮＥＴコード', '提出者種別', '上場区
分', '連結の有無', '資本金', '決算日', '提出者名', '提出者名（英字）', '提出者名（ヨ
ミ）', '所在地', '提出者業種', '証券コード'))

df2 = df1.drop([0,1])
df3 = df2['ＥＤＩＮＥＴコード'].reset_index(drop=True)

df3.to_csv('EDINETcode.csv',index=False)
```

リスト 2

```
import pandas as pd
import codecs

df_j1 = pd.read_excel('data_j.xlsx', 'Sheet1', index_col=None)
df_j2 = df_j1['コード']

with codecs.open('EdinetcodeDlInfo.csv', "r", "Shift-JIS", "ignore")
 as file:
    df_e1 = pd.read_csv(file, names=('ＥＤＩＮＥＴコード', '提出者種別',
'上場区分', '連結の有無', '資本金', '決算日', '提出者名', '提出者名（英字）', '提
出者名（ヨミ）','所在地','提出者業種','証券コード'))

df_e2 = df_e1.drop([0,1]).reset_index(drop=True)
```

```
i_se = pd.Series()
for i in df_j2:
    i_ecode = df_e2[df_e2['証券コード'] == str(i*10)]['ＥＤＩＮＥＴコード
']
    if len(i_ecode) == 0: continue
    i_se = i_se.append(i_ecode)

i_se2 = i_se.reset_index(drop=True)
i_se2.to_csv('TD_Ecode.csv', index=False)
```

また，EDINET提供のリストの始め2行は不要であるかつリストの形が均等にならないことから，“drop([0, 1])”を用いて削除している。“to_csv”はCSVファイル形式でファイル出力するコマンドである。“str(i*10)”は［証券コード×10］を表している。これは，EDINETのリストでは証券コードの値が，東京証券取引所で用いられている証券コードに10倍した値が用いられているためである[6]。

【リスト2】では，東京証券取引所からダウンロードした証券コードのリストを読み込み，証券コードのみの列を作成している。そしてその証券コードと一致するEDINETコードを抽出し，EDINETコードを得ている。

結果，【リスト1】から得られるEDINETコードは9,725件（2018年7月8日版），【リスト2】から得られるEDINETコードは3,595件（2018年6月末日版）であった。証券コードを利用した【リスト2】では，東京証券取引所に上場している企業に限られるため，件数が少なくなっている。一方で，【リスト1】から得られたEDINETコードの大半は有価証券報告書を提出しない提出者であるため，得られる有価証券報告書の数はEDINETコード数の差ほどは違わない。しかし，サントリーホールディングス株式会社といった，上場していない大企業は数多くあるためそちらが含まれないことに注意しなければな

6）その理由は不明である。

らない。

第 3 節　EDINET から XBRL ファイルのダウンロード

EDINET コードのリストが準備できれば，次は EDINET にアクセスすることになる。EDINET は JavaScript で制御されているため，URL を指定して【図 1】のような企業のページにアクセスすることができない。【図 1】にも URL は存在しているが，検索をする毎に一時的な URL が与えられるため，同じ URL でアクセスすることができない仕様になっている。本稿では，ウェブブラウザを自動操作する方法を用いて EDINET からファイルをダウンロードする。

JavaScript を使った頁からスクレイピングするためには，JavaScript を解釈できるクローラーが必要になる。本稿では Selenium と呼ばれるクローラーを使用する。Selenium は様々なブラウザーを自動操作するツールである。元々 Web アプリケーションの自動テストツールとして発展したが，JavaScript を使ったページからスクレイピングするためにも使うことができる。ブラウザーを操作するためのドライバーが用意されており，ドライバー経由で様々なブラウザーを操作できる[7]。

図 1　EDINET の検索結果

○検索結果

14 件中(1 ～ 14 件表示)　　XBRL一括ダウンロード

提出日時	提出書類	コード	提出者／ファンド	発行／対象／子会社／臨報提出事由	PDF	XBRL	比較	備考
H30.06.19 15:45	有価証券報告書－第101期(平成29年4月1日－平成30年3月31日)	E01777	ソニー株式会社		PDF	XBRL	COMPARE	
H30.02.08 15:00	四半期報告書－第101期第3四半期(平成29年10月1日－平成29年12月31日)	E01777	ソニー株式会社		PDF	XBRL	COMPARE	
H29.11.07 15:01	四半期報告書－第101期第2四半期(平成29年7月1日－平成29年9月30日)	E01777	ソニー株式会社		PDF	XBRL	COMPARE	
H29.08.07 15:02	四半期報告書－第101期第1四半期(平成29年4月1日－平成29年6月30日)	E01777	ソニー株式会社		PDF	XBRL	COMPARE	

【リスト1】から作成されたEDINETコードのリストから各企業（EDINETコード）を選択し，対応する企業のXBRLファイルをダウンロードするコードが【リスト3】である。このコードに関しては，【リスト1】の内容が含まれており，EDINETからダウンロードした“EdinetcodeDlInfo.csv”があれば動作するように作成している。

“#ヘッドレスモードにする”コードを有効にすると，コード利用時にブラウザーが立ち上がらず，モニターに見えない状態で処理が行われる。“URLの入力”では，EDINETの書類詳細検索画面を指定する。EDINETの書類詳細検索画面では，「XBRLを含む書類のみを検索対象とする」のチェックを付ける項目がある。さらに，「書類提出者情報を指定する」，「有価証券発行者情報を指定する」，「ファンド情報を指定する」，「書類種別を指定する」，「決算期／提出期間を指定する」の5項目を指定し検索を行う。Seleniumにはチェックボックスやボタンクリックの動作を操作する機能があるが，指定するチェックボックスの要素を調べる必要がある。

例えば，「XBRLを含む書類のみを検索対象とする」の項目は，下記のようなタグで記述されており，id要素の“xbr_c”が必要となる。これらタグはページのソースを表示させて調べるという作業をしなければならない。また，書類詳細検索画面にアクセスしたとき，「書類提出者情報を指定する」のウィンドウは開いた状態で始まるが，他の「有価証券発行者情報を指定する」，「ファンド情報を指定する」，「書類種別を指定する」，「決算期／提出期間を指定する」の4項目は閉じた状態であるため，クリックしてウィンドウを開かなくてはならない。これらを指定する要素は，「XBRLを含む書類のみを検索対象とする」の項目の“xbr_c”のように，要素名が分かりやすい名称とはなっていない。“a”，“b”，“c”といった文字列や序数を用いることが多く，実際に目で見て確認しなければ特定することができない。

7）加藤［2017］，191頁。

```
<input type="checkbox" name="xbr" id="xbr_c" value="on">
```

「書類種別を指定する」の項目では，有価証券報告書，四半期報告書，半期報告書の３つの項目を指定する。この作業をしなければ，有価証券届出書や臨時報告書といった書類が検索結果に表示されることになる。これらを用いるのであれば問題はないが，一般的によく用いる有価証券報告書や四半期報告書を対象とする場合，あらかじめ指定する方が効率が良い。

「決算期／提出期間を指定する」の項目では，決算期や提出期間を任意に指定する。何も指定しなければ，過去１年間に固定されており，複数年を対象として検索ができない。なお，EDINET では，有価証券報告書は最新のものを含む過去５年間，四半期報告書および半期報告書は最新のものを含む過去３年間が保管されている。

必要な項目を指定した後，任意の EDINET コードを「書類提出者情報を指定する」のウィンドウの提出者 EDINET コードに入力し検索を行う。検索結果は【図１】と同様の画面が表示され，「XBRL 一括ダウウンロード」のボタンをクリックするとダウウンロードが始まる。

【リスト３】には数秒待機させるコマンドを数か所含めている。ダウンロード中などの処理が終えていないにもかかわらず，次の処理を行うことを防ぐためである。EDINET コードは存在しているが，XBRL ファイルを含む書類を提出していない企業も多く存在する。このような場合にはエラーが生じるため，エラーを回避するような仕組みも取り入れなければならない。このように，処理を完了させるために試行錯誤を重ねる必要がある。

リスト３

```
import os
import time
import pandas as pd
import codecs
```

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import Select
from selenium.webdriver.common.alert import Alert
from selenium.common.exceptions import NoSuchElementException

#オプションの設定
options = Options()

#Chrome canary のある場所（.exe まで指定）
options.binary_location = 'Chrome canary へのパス'

#ヘッドレスモードにする
#options.add_argument('--headless')

#ダウンロードフォルダを指定する
prefs = {'download.default_directory' : 'ダウンロードフォルダへのパス'}
options.add_experimental_option('prefs', prefs)

#ChromeDriver のある場所（.exe まで指定）
driver = webdriver.Chrome(chrome_options=options, executable_path='
ChromeDriver へのパス')

#URL の入力
driver.get('https://disclosure.edinet-
fsa.go.jp/E01EW/BLMainController.jsp?uji.bean=ee.bean.parent.EECom
monSearchBean&uji.verb=W0EZA230CXP002010BLogic&TID=W1E63020&PID=W0
EZ0001&SESSIONKEY=&lgKbn=2&dflg=0&iflg=0')

#「XBRL を含む書類のみを検索対象とする」をチェック
check_element = driver.find_element_by_id('xbr_c')
check_element.click()

#決算期/提出機関のウィンドウを開く
```

```
form_class = 'panel-title-blue'
term_window = driver.find_elements_by_class_name(form_class)[3]
term_window.click()
time.sleep(2)

#「書類種別を指定する」をクリック
formclass_button = driver.find_element_by_id('doc_kn2')
formclass_button.click()

#「有価証券報告書」「四半期報告書」「半期報告書」にチェック
form_otd1_button = driver.find_element_by_id('otd1')
form_otd1_button.click()
time.sleep(1)
form_otd3_button = driver.find_element_by_id('otd3')
form_otd3_button.click()
time.sleep(1)
form_otd5_button = driver.find_element_by_id('otd5')
form_otd5_button.click()
time.sleep(1)

#決算期/提出機関のウィンドウを開く
term_class = 'panel-title-blue'
term_window = driver.find_elements_by_class_name(term_class)[4]
term_window.click()
time.sleep(2)

#期間を全期間に選択する
term_element = driver.find_element_by_name('pfs')
Select(term_element).select_by_value('5')
time.sleep(2)

#検索ボタンをクリックする
search_button5 = driver.find_element_by_name('forward_sch5')
search_button5.click()
time.sleep(5)
```

```
def main() :
    with codecs.open('EdinetcodeDlInfo.csv', "r", "Shift-JIS",
"ignore") as file:
        df1 = pd.read_csv(file, names= ('ＥＤＩＮＥＴコード','提出者種別',
'上場区分', '連結の有無', '資本金', '決算日', '提出者名', '提出者名（英字)', '提出
者名（ヨミ)', '所在地', '提出者業種', '証券コード'))

    df2 = df1.drop([0,1])
    df3 = df2['ＥＤＩＮＥＴコード']

    #edinet_code = ['E00045', 'E03144', 'E09691', 'E01772']
    #noXBRL_list = list()
    for i in df3:
        #「書類提出者情報を指定する」のウィンドウを開く
        term_class = 'panel-title-blue'
        term_window
driver.find_elements_by_class_name(term_class)[0]
        term_window.click()
        time.sleep(2)

        code = xbrldown(i)
        if code != None:
            fi = open('NoXBRL.txt', 'a', encoding='utf-8')
            print(code)
            print(code, file=fi)
            fi.close()

def xbrldown(edinet_code):
    #EDINET コードを入力する
    input_element = driver.find_element_by_name('sec')
    input_element.clear()
    input_element.send_keys(edinet_code)

    #検索ボタンをクリックする
    search_button1 = driver.find_element_by_id('sch')
```

```
    search_button1.click()

    #XBRL の一括ダウンロードボタンをクリックする
    try:
        download_button = driver.find_element_by_name('forward_xbrl')
    except NoSuchElementException:
        return edinet_code
    else:
        download_button.click()

    #アラートで「OK」を選択する
    Alert(driver).accept()
    time.sleep(3)

    #ダウンロード中かどうかを判断する
    start = os.getcwd()
    os.chdir('ダウンロードフォルダへのパス')

    while True:
        ext_list=list()
        for i in os.listdir():
            #global ext
            ext = os.path.splitext(i)
            ext_list.append(ext[1])
        time.sleep(3)
        if '.crdownload' not in ext_list: break

    os.chdir(start)
    return

if __name__ == '__main__':
    main()

driver.quit()
```

第4節 XBRLファイルの解凍とXBRLファイルからのデータ抽出

ダウンロードしたXBRLファイルはZIPファイルと呼ばれる圧縮された状態で保存されているため，それぞれ解凍しなければデータを抽出することはできない。さらに，PDFファイルとは異なり，XBRL形式の報告書は複数のファイルから構成されている。したがって，XBRLファイルを解凍し，該当するXBRLファイルのみを保存する処理が求められる。解凍するコードを【リスト4】に示す。【リスト4】では，ダウンロードした圧縮フォルダを，任意のフォルダに全て解凍するようになっている。

圧縮ファイルは企業毎にまとめられて保存されている。中身は各報告書がそれぞれフォルダに分けられており，さらに一つのCSVファイルで構成されている。例えば，対象期間で企業が2つの有価証券報告書と6つの四半期報告書を提出していた場合，圧縮ホルダには8つの報告書ファイルと1つのCSVファイルが含まれることになる。報告書のファイルの名称は書類管理番号と呼ばれる“S100AFIH”といった文字の羅列になっており，名称からはどの企業の何の報告書かはわからない。CSVファイルには，それぞれの報告書と書類管理番号が【表2】のように対応づけられている。

表2

書類管理番号	書類名	EDINET（ファンド）コード	提出者（ファンド）
S100AFIH	有価証券報告書	E01777	ソニー株式会社

【表2】の通り，CSVファイルに含まれる項目は「書類管理番号」,「書類名」,「EDINET（ファンド）コード」,「提出者（ファンド）」の4つであり，どの会計期間の有価証券報告書なのかは各報告書ファイルに含まれるファイルを確認しなければわからないようになっている。書類管理番号をキーとして決算日や各種勘定科目の金額のデータを揃えることになる。そのため，各報告書のフォルダ名を書類管理番号とし，企業フォルダの名称はEDINETコードに

する。例えば，ソニー株式会社のフォルダ名は“E01777”となり，その下に“S100AFIH”のフォルダが置かれ，そのフォルダは有価証券報告書の XBRL ファイルが含まれることになる。

リスト 4

```
import os
import shutil
import zipfile
import codecs
import glob
import pandas as pd

#ZIP ファイルがあるフォルダ
path_zipTop = 'ダウンロードしたフォルダへのパス'
path_zip = []
path_zip = glob.glob(path_zipTop)

dir_top = os.getcwd()

for i in path_zip:

    #ZIP ファイルを解凍するフォルダ
    os.chdir('解凍するフォルダへのパス')

    with zipfile.ZipFile(i) as existing_zip:
        existing_zip.extract('XbrlSearchDlInfo.csv')

    with codecs.open('XbrlSearchDlInfo.csv', 'r', 'Shift-JIS',
'ignore') as file:
        df1 = pd.read_csv(file, header=1)

    #EDINET コードを取得する（例：E03144）
    df2 = df1['EDINET(ファンド)コード'][0]
```

```
    #新しいフォルダを作成し，フォルダ名を EDINET コードにする
    os.chdir('D:¥¥XBRL2¥¥XBRL_FILE')
    os.mkdir(df2)

    dir_start = os.getcwd()
    file_path = os.path.join(dir_start, df2)

    #フォルダの中身を全て解凍して展開
    with zipfile.ZipFile(i) as existing_zip:
        existing_zip.extractall(file_path)

    #「書類名」に「訂正」が含まれるフォルダを削除
    df3 = df1[df1['書類名'].str.contains('訂正')]['書類管理番号']
    if len(df3) > 0:
        for j in df3:
            shutil.rmtree('解凍するフォルダへのパス' + df2 + '¥¥' + j)

    print(df2 + ' is OK !')

    file_path2 = os.path.join(dir_start, 'XbrlSearchDlInfo.csv')
    os.remove(file_path2)
    os.chdir(dir_top)

os.chdir('D:¥¥XBRL2¥¥XBRL_FILE')

#フォルダ名が「G～」となっているものを削除
df1 = pd.DataFrame(os.listdir('D:¥¥XBRL2¥¥XBRL_FILE'))
df1.columns = ['ＥＤＩＮＥＴコード']

df2 = df1[df1['ＥＤＩＮＥＴコード'].str.contains('G')]['ＥＤＩＮＥＴコード
']

for i in df2:
    shutil.rmtree('解凍するフォルダへのパス' + i)
```

```
df3 = pd.DataFrame(os.listdir('解凍するフォルダへのパス'))
df3.columns = ['ＥＤＩＮＥＴコード']
```

【リスト１】や【リスト２】で作成した EDINET コードでダウンロードしたファイルには，一部の信託などのファイルが含まれてしまい，信託などの有価証券報告書が含まれる。一般に，これらのデータは会計分野の分析対象には含まないことから，【リスト４】の後半部分において信託等のファイルを削除する処理を含めている。

ここで，XBRL 化された報告書のファイル構成について見ていく。“S100AFIH”のような報告書フォルダを展開すると，“AuditDoc”と“PublicDoc”の２つのフォルダがある。前者が監査報告書であり，後者が提出本文書である。提出本文書には，図表のフォルダ，複数の HTML 文書，複数の XML 文書，１つの XBRL ファイル，１つの XSD ファイルがある。データ抽出のために用いるファイルは XBRL ファイルである。

次に，XBRL ファイルからデータを抽出する。本稿では例として，有価証券報告書のデータの中から，報告書の期末日および当期の親会社株主に帰属する当期純利益を取り出してみる。ここでは，決算日が 2015 年，2016 年，2017 年に含まれる企業を対象とする。決算日の集計結果を示したものが【表３】である。

【表３】の特徴として，日本の企業の決算日は月末に集中しているということである。米国企業は，多少の偏りがあるものの特定の日を除く１年 365 日に分散している。日本企業で最も多いのは，年度末の３月 31 日であり，6.5 割前後の企業がこの日を決算日としている。次に多いのが年末の 12 月 31 日であり，約１割の企業がこの日を決算日としている。また，２月末の企業が約５％あることは興味深い結果となった。

次に企業が採用する会計基準の集計結果を【表４】に示す。依然として日本の会計基準を採用する企業が多いことがわかる。また米国会計基準（US-GAAP）

を採用する企業は年々減少している。中溝（2017）で指摘しているが，一部の企業が米国会計基準からIFRSへの移行を行っている背景がある。また，内閣府が目指すIFRS採用企業300社には到底届いていない現状が窺える。

表3 決算日の集計（2015～2017年）

2015			2016			2017		
合 計	4,054	100%	合 計	4,065	100%	合 計	4,089	100%
2015-01-20	3	0.07%	2016-01-20	3	0.07%	2017-01-20	3	0.07%
2015-01-31	53	1.31%	2016-01-31	55	1.35%	2017-01-31	55	1.35%
2015-02-15	1	0.02%	2016-02-15	1	0.02%	2016-02-15	0	0.00%
2015-02-20	15	0.37%	2016-02-20	13	0.32%	2017-02-20	12	0.29%
2015-02-28	198	4.88%	2016-02-29	200	4.92%	2017-02-28	207	5.06%
2015-03-15	2	0.05%	2016-03-15	2	0.05%	2017-03-15	2	0.05%
2015-03-20	19	0.47%	2016-03-20	19	0.47%	2017-03-20	19	0.46%
2015-03-25	1	0.02%	2016-03-25	1	0.02%	2017-03-25	1	0.02%
2015-03-31	2,725	67.22%	2016-03-31	2,701	66.45%	2017-03-31	2,680	65.54%
2015-04-20	2	0.05%	2016-04-20	2	0.05%	2017-04-20	2	0.05%
2015-04-30	37	0.91%	2016-04-30	40	0.98%	2017-04-30	41	1.00%
2015-05-15	3	0.07%	2016-05-15	3	0.07%	2017-05-15	3	0.07%
2015-05-20	5	0.12%	2016-05-20	5	0.12%	2017-05-20	4	0.10%
2015-05-31	74	1.83%	2016-05-31	74	1.82%	2017-05-31	73	1.79%
2015-06-20	4	0.10%	2016-06-20	4	0.10%	2017-06-20	3	0.07%
2015-06-30	119	2.94%	2016-06-30	126	3.10%	2017-06-30	131	3.20%
2015-07-20	2	0.05%	2016-07-20	2	0.05%	2017-07-20	2	0.05%
2015-07-31	36	0.89%	2016-07-31	36	0.89%	2017-07-31	37	0.90%
2015-08-20	3	0.07%	2016-08-20	3	0.07%	2017-08-20	3	0.07%
2015-08-31	65	1.60%	2016-08-31	67	1.65%	2017-08-31	69	1.69%
2015-09-20	4	0.10%	2016-09-20	3	0.07%	2017-09-20	3	0.07%
2015-09-30	161	3.97%	2016-09-30	165	4.06%	2017-09-30	172	4.21%
2015-10-20	2	0.05%	2016-10-20	2	0.05%	2017-10-20	2	0.05%
2015-10-31	44	1.09%	2016-10-31	45	1.11%	2017-10-31	46	1.12%
2015-11-20	2	0.05%	2016-11-20	2	0.05%	2017-11-20	2	0.05%
2015-11-30	53	1.31%	2016-11-30	56	1.38%	2017-11-30	57	1.39%
2015-12-20	5	0.12%	2016-12-20	4	0.10%	2017-12-20	3	0.07%
2015-12-31	416	10.26%	2016-12-31	431	10.60%	2017-12-31	457	11.18%

表 4　採用会計基準と黒字企業の割合

	2015		2016		2017	
合　計	4,054	100%	4,065	100%	4,089	100%
日本基準	3,966	97.83%	3,973	97.74%	3,970	97.09%
米国基準	27	0.67%	24	0.59%	20	0.49%
IFRS	61	1.50%	67	1.65%	98	2.40%
要素エラー	0	0.00%	1	0.02%	1	0.02%
黒字企業	3,569	88.04%	3,570	87.82%	3,669	89.73%

黒字企業の割合は約 9 割であることが判明した。中小企業を含めた場合，赤字企業の割合が多くなるが，EDINET に報告書を提出する企業に限って言えば，9 割の企業が黒字である。要素エラーは，1 社のみ XBRL の要素名をその企業独自のものを使用していたことが原因である。

第 5 節　お　わ　り　に

本稿では，プログラムを使用して，EDINET に提出された XBRL 形式の会計報告書を自動でダウンロードし，データを抽出することを試みた。範囲は限定的であるが，これまでのデータベースでは得ることのできなかった情報も取得できることがわかった。会計研究の分野においても，今後は既存のデータベースに依存するだけでなく，必要なデータを収集し，分析を行うことが求められるだろう。特に，坂上（2017）で取り上げられているような機械学習の技術を用いた研究では，本稿のようなデータ収集をすることが必要となる。

一方で，プログラムを用いてデータを収集するには試行錯誤を繰り返すことになる。EDINET 上のデータ数のようなデータが公表されていないため，取得したデータが予定通りのデータを取得できているかどうかを確かめることは難しい。信頼性のあるデータセットを作成するために，データセットを評価する視点を欠かすことはできない。これからのデータ分析を行う上で考えていかなくてはならない。

参 考 文 献

加藤耕太［2017］,『Pythonクローリング＆スクレイピング－データ収集・解析のための実践開発ガイド－』技術評論社。

小谷 融［2007］,「EDINETの現状と課題」河﨑照行（編著）『電子情報開示のフロンティア』中央経済社，第4章，43-54頁。

坂上 学［2011］,『会計人のためのXBRL入門 新版』同文舘。

――――［2017］,「会計研究におけるディープラーニングの適用可能性」産業経理，第77巻第3号，84-97頁。

使用したPython環境

Python 3. 6. 4
beautifulsoup4 == 4. 6. 0
certify == 2018. 1. 18
chardet == 3. 0. 4
cycler == 0. 10. 0
idna == 2. 6
lxml == 4. 1. 1
matplotlib == 2. 1. 2
numpy == 1. 14. 0
pandas == 0. 22. 0
pyparsing == 2. 2. 0
python-dateutil == 2. 6. 1
pytz == 2017. 3
requests == 2. 18. 4
scipy == 1. 0. 0
seaborn == 0. 8. 1
selenium == 3. 8. 1
six == 1. 11. 0
urllib3 == 1. 22
xlrd == 1. 1. 0