e-mentor
DWUMIESIĘCZNIK SZKOŁY GŁÓWNEJ HANDLOWEJ W WARSZAWIE
WSPÓŁWYDAWCA: FUNDACJA PROMOCJI I AKREDYTACJ KIERUNKÓW EKONOMICZNYCH

2016, nr 2 (64)

J. Swacha, *Analiza graficzna danych edukacyjnych z wykorzystaniem języka Python*, „e-mentor” 2016, nr 2(64), s. 13–21, http://dx.doi.org/10.15219/em64.1239.

Analiza graficzna danych edukacyjnych z wykorzystaniem języka Python

Jakub Swacha

Choć analiza danych edukacyjnych kojarzona jest często ściśle z gigadanymi, bardzo wartościowe źródło stanowią dla niej niewielkie zbiory danych edukacyjnych, dostępne każdemu nauczycielowi. Do badania zbiorów o niewielkim rozmiarze szczególnie dobrze nadaje się analiza graficzna, która w ich przypadku jest w stanie dostarczyć wyniki nie tylko przejrzyste, ale także precyzyjne. W niniejszym opracowaniu posłużono się demonstracyjnym studium przypadku, aby ukazać przydatność i łatwość wykorzystania na tym polu języka programowania Python.

W ostatnich latach wiele uwagi w obszarze technologii informacyjno-komunikacyjnych poświęca się fenomenowi gigadanych (*big data*), którego pojawienie się przyrównuje się wręcz do „szóstej fali rewolucji komputerowej”[1]. Choć gigadanymi zainteresował się w pierwszej kolejności biznes[2], mają one dużą wartość także dla edukacji, o czym świadczy szybki rozwój analityki uczenia się (*learning analytics*) i eksploracji danych edukacyjnych (*educational data mining*)[3].

O ile eksploracja danych edukacyjnych nieodłącznie związana jest z gigadanymi, o tyle analityka uczenia się może być realizowana także w oparciu o niewielkie zbiory danych[4]. Takiego podejścia nie należy traktować jako anachronicznego: do jego zalet należy zaliczyć zarówno dostępność danych, jak i to, że dotyczą one precyzyjnie wybranych obiektów w ściśle określonym zakresie. R. Kitchin i T.P. Lauriault ujmują to za pomocą obrazowej metafory, że o ile badania oparte na gigadanych przypominają wielką kopalnię odkrywkową, w której pozyskuje się rudę, przekopując olbrzymie zwały ziemi, o tyle badania wykorzystujące niewielkie zbiory danych odpowiadają klasycznej kopalni, w której drąży się wąski chodnik śladem pokładu wartościowego minerału[5].

Co więcej, podejście oparte na analizie niewielkich zbiorów danych wolne jest od wad specyficznych dla gigadanych, w szczególności wysokich wymagań względem niezbędnej infrastruktury technicznej oraz ryzyka naruszenia prawa do prywatności uczących się osób, wynikającego zarówno z faktu zbierania szerokiego spektrum danych mających stanowić kontekst analityczny, jak i ryzyka nieuprawnionego dostępu do zebranych już danych, które wiąże się z przechowywaniem i komputerowym przetwarzaniem dużych zbiorów danych z wykorzystaniem publicznych chmur obliczeniowych[6].

Za szczególnie przydatną do analizy niewielkich zbiorów danych edukacyjnych należy uznać analizę graficzną, ze względu na to, że zbiory takie mogą być wizualizowane w całości i jednocześnie z wysoką precyzją. Dodatkową jej zaletą jest łatwość posługiwania się nią, co czyni ją dostępną także dla osób o ograniczonej wiedzy statystycznej.

Analiza danych edukacyjnych, niezależnie od skali, wykonywana jest z wykorzystaniem narzędzi informatycznych, które zapewniają możliwość manipulacji kolekcjami danych, dostęp do szerokiego wyboru funkcji statystycznych i wizualizację danych w formie dostosowanej do specyficznych potrzeb. Co

[1] P. Płoszajski, *Big Data: nowe źródło przewag i wzrostu firm*, „e-mentor” 2013, nr 3(50), s. 5–10, http://www.e-mentor.edu.pl/artykul/index/numer/50/id/1016.

[2] Zob. np. I. Miciuła, K. Miciuła, *Kluczowe trendy dla budowania biznesu w branży big data*, „Zeszyty Naukowe Uniwersytetu Szczecińskiego. Studia Informatica” 2015, nr 36(863), s. 51–63, http://dx.doi.org/10.18276/si.2015.36-04.

[3] Z. Papamitsiou, A. Economides, *Learning Analytics and Educational Data Mining in Practice: A Systematic Literature Review of Empirical Evidence*, „Educational Technology & Society” 2014, Vol. 17, No. 4, s. 49–64, http://www.jstor.org/stable/jeductechsoci.17.4.49.

[4] Zob. np. S. Goggins, W. Xing, X. Chen, B. Chen, B. Wadholm, *Learning Analytics at „Small” Scale: Exploring a Complexity-Grounded Model for Assessment Automation*, „Journal of Universal Computer Science” 2015, Vol. 21, No. 1, s. 66–92, http://www.jucs.org/jucs_21_1/learning_analytics_at_small, [22.04.2016].

[5] R. Kitchin, T.P. Lauriault, *Small data in the era of big data*, „GeoJournal” 2015, Vol. 80, No. 4, s. 466, http://dx.doi.org/10.1007/s10708-014-9601-7.

[6] Zob. np. A. Gross, *A brief history of education's big data debate*, „Education Dive”, 07.05.2014, http://www.educationdive.com/news/a-brief-history-of-educations-big-data-debate/258602.

najmniej kilka takich narzędzi oparto na interpreterze języka programowania Python. Język ten, z racji swej prostoty, przejrzystości i produktywności, cieszy się dużym zainteresowaniem specjalistów z różnych obszarów[7].

Głównym celem niniejszego opracowania jest zatem zademonstrowanie przydatności: analizy graficznej do analizy danych edukacyjnych (w szczególności dotyczącej niewielkich zbiorów danych), a języka Python do analizy graficznej. Wśród celów szczegółowych wyróżnić można:

- wskazanie narzędzi informatycznych ułatwiających korzystanie z języka Python,
- pokazanie, jak nietypowa forma przeprowadzenia testu poszerza kontekst analizy jego wyników,
- wykazanie zdolności języka Python do precyzyjnego dostosowania formy wizualizacji danych do specyficznych potrzeb,
- wykazanie zdolności języka Python do szybkiej wizualizacji danych,
- porównanie możliwości środowiska analizy danych edukacyjnych opartego na języku Python do możliwości środowiska opartego na programie Microsoft Excel.

Jako podstawową metodą badawczą posłużono się demonstracyjnym studium przypadku.

Wykorzystane oprogramowanie

W przeprowadzonym studium przypadku zdecydowano się wykorzystać pakiet Anaconda, rozwijany przez firmę Continuum Analytics. Wyboru tego dokonano, mając na uwadze następujące jego zalety:

- dostępność bezpłatnych wersji przeznaczonych dla różnych systemów operacyjnych (Windows, Linux, OS X), w tym także obrazów maszyn wirtualnych przygotowanych dla różnych dostawców usług chmurowych IaaS (m.in. Amazon AWS i Microsoft Azure),
- dostępność „od zaraz" w formule usługi chmurowej SaaS[8],
- interaktywną powłokę interpretera IPython udostępnioną za pośrednictwem notatnika Jupyter, pozwalającą na tworzenie hybrydowych dokumentów złożonych z komórek zawierających m.in. hipertekst i kod źródłowy, który może być wykonywany bezpośrednio w środowisku przeglądarki internetowej[9],
- narzędzie Conda znacząco ułatwiające zarządzanie modułami[10],
- ponad 300 dołączonych specjalistycznych bibliotek.

Źródło i specyfika analizowanych danych

Do analizy wybrano autentyczne wyniki testu zaliczeniowego z podstaw programowania komputerów przeprowadzonego wśród studentów I roku kierunku *informatyka i ekonometria* na Wydziale Nauk Ekonomicznych i Zarządzania Uniwersytetu Szczecińskiego (rok akademicki 2012/2013). Analizie poddano 48 wypełnionych formularzy testowych, które na potrzeby publikacji zanonimizowano.

Zasady przeprowadzenia testu były następujące:

- blok główny stanowiło 25 pytań, dla każdego podano cztery warianty odpowiedzi,
- zawsze tylko jeden wariant był prawidłowy, a jeden wśród nieprawidłowych („bardzo zły") zawierał odpowiedź nielogiczną bądź taką, której wybór świadczył o tym, że student nie dysponuje podstawową wiedzą z przedmiotu,
- studenci mogli zaznaczyć jedną, dwie lub trzy odpowiedzi i otrzymać odpowiednio jeden, pół lub jedną trzecią punktu, jeżeli wśród wybranych wariantów był prawidłowy,
- jeżeli wśród wybranych wariantów był „bardzo zły", odbierano jeden punkt (niezależnie od liczby wybranych wariantów),
- zaznaczenie wszystkich czterech wariantów traktowano jako brak odpowiedzi (0 punktów).

Możliwość wyboru więcej niż jednej odpowiedzi wprowadzono po to, by móc ocenić pewność studentów co do własnej wiedzy. „Bardzo złe" warianty odpowiedzi miały zaś zmotywować ich do głębszego przemyślenia treści nawet tych pytań, na które nie znali prawidłowej odpowiedzi, i albo wskazania trzech odpowiedzi uznanych za prawdopodobne, albo zupełnego odstąpienia od odpowiedzi na dane pytanie.

Cele i metoda analizy wyników testu

Za główne cele przeprowadzenia analizy wyników testu należy uznać:

1) ocenę jakości samego testu, i – w konsekwencji – udoskonalenie go poprzez zmodyfikowanie pytań zbyt łatwych lub zbyt trudnych oraz zadbanie o możliwie pełne i równomierne pokrycie pytaniami zarówno zakresu merytorycznego wykładów, jak i oczekiwanych efektów kształcenia;
2) ocenę doboru i formy przekazywanych treści dydaktycznych, stanowiących podstawę poszczególnych pytań ujętych w teście poprzez identyfikację zakresu merytorycznego treści, z którego opanowaniem studenci szczególnie

[7] Zob. np. J.M. Eppler, M. Helias, E. Muller, M. Diesmann, M.-O. Gewaltig, *PyNEST: A Convenient Interface to the NEST Simulator*, „Frontiers in Neuroinformatics" 2008, No. 2, art. 12, http://dx.doi.org/10.3389/neuro.11.012.2008.
[8] *Wakari.io. Web-based Python Data Analysis*, http://wakari.io, [20.03.2016].
[9] C. Rossant, *Learn IPython for interactive Python programming, high-performance numerical computing, and data visualization*, Packt Publishing, Birmingham 2013, s. 38.
[10] M. Gorelick, I. Ozsvald, *High Performance Python: Practical Performant Programming for Humans*, O'Reilly Media, Sebastopol 2014.

sobie nie poradzili, i – w konsekwencji – udoskonalenie materiałów dydaktycznych poprzez np. jaśniejsze sformułowanie przekazu czy dodanie przykładów lub ćwiczeń do samodzielnego wykonania;

3) wyjaśnienie przyczyn uzyskania przez poszczególnych studentów określonych wyników, w szczególności: zbadanie zależności między faktem uczenia się (np. obecność na określonych wykładach) a uzyskanymi w teście wynikami oraz wykrycie zbieżności pomiędzy odpowiedziami różnych studentów.

W dalszej części artykułu przedstawione zostaną ukierunkowane na powyższe cele przykłady analizy graficznej wyników testu przeprowadzonej z wykorzystaniem oprogramowania Anaconda, opartego na języku Python. Analiza obejmowała zbadanie: poprawności i pewności odpowiedzi udzielonych na poszczególne pytania, zależności wyników od obecności na wykładach, rodzaju pytania i zakresu treści, których ono dotyczy, oraz stopnia podobieństwa odpowiedzi udzielanych przez różnych studentów.

Przykłady wykorzystania języka Python do analizy wyników testu

Niezależnie od tego, czy test przeprowadzono elektronicznie, czy tylko wprowadzono do komputera jego wyniki (jak było to w omawianym przypadku), prostą i wygodną formę ich reprezentacji stanowi plik tekstowy zawierający umieszczone w kolejnych wierszach (odpowiadających poszczególnym studentom) ciągi wartości (odpowiadające poszczególnym pytaniom) rozdzielone przecinkami. Kod wczytujący tabelę wyników i wydzielający z niej do osobnych tabel informacje zawarte w dodatkowych kolumnach (obecności studentów na kolejnych wykładach) i wierszach (kolejno: prawidłowe i „bardzo złe" warianty odpowiedzi, rodzaj pytania oraz numer wykładu, którego dotyczyło) zawiera listing 1.

Analiza odpowiedzi na poszczególne pytania testowe

Dość złożone reguły testu skomplikowały sprawdzanie poprawności i pewności udzielonych odpowiedzi (patrz listing 2). Mimo to warto zwrócić uwagę np. na prostotę zapisu operacji przemnożenia wszystkich danych w kolekcji przez skalar (tu: 100.0/N).

Uzyskanie specyficznego wyglądu histogramu (m.in. wyświetlenie liczby trafnych odpowiedzi powyżej, a nietrafnych – poniżej osi poziomej) wymaga kilkunastu wierszy kodu (patrz listing 3). Rezultat jego wykonania[11] przedstawiono na rysunku 1.

Analizując diagram, można zauważyć przede wszystkim brak poprawnych odpowiedzi na pytanie 14. Również pytania 16, 18 i 23 okazały się dla zdecydowanej większości studentów zbyt trudne. Z kolei pytanie 3 okazało się zbyt proste, podobnie

Listing 1. Wczytanie pliku CSV

```
import pandas as pd
wyniki = pd.read_csv(r"..\wyniki_testu.csv", sep=',',
         encoding='iso8859_2', na_filter=False, index_col=0)
N = 48;   P = 25;   W = 7   # liczba studentów, pytań, wykładów
obecnosci = wyniki.iloc[0:N,P+1:P+W+1].astype(int)
wyniki.drop(wyniki.columns[P:], axis=1, inplace=True)
prawidlowe = wyniki.iloc[N];    najgorsze = wyniki.iloc[N+1]
rodzaj_pytania = wyniki.iloc[N+2];
numer_wykladu = wyniki.iloc[N+3].astype(int)-1;   wyniki = wyniki[:-4]
```

Źródło: opracowanie własne.

Listing 2. Sprawdzanie poprawności i pewności udzielonych odpowiedzi

```
import numpy as np
def punktacja(odp, dobra, zla):
    if odp == dobra:  return 1.0      # pewna dobra
    elif zla in odp:  return -1.0     # bardzo zła
    if not 1<len(odp)<4 or dobra not in odp:  return 0.0 # zła
    else:   return 1.0/len(odp)       # niepewna dobra
oceny = np.array(map(lambda pytanie, dobra, zla: map(lambda pytanie:
        punktacja(pytanie, dobra, zla), wyniki[pytanie]),
        list(wyniki), prawidlowe, najgorsze))
oceny_pytan = (100.0 / N) * np.swapaxes(np.apply_along_axis(lambda x:
   np.histogram(x, bins=[-2.0,-0.1,0.1,0.9,1.1])[0], 1, oceny), 0, 1)
```

Źródło: opracowanie własne.

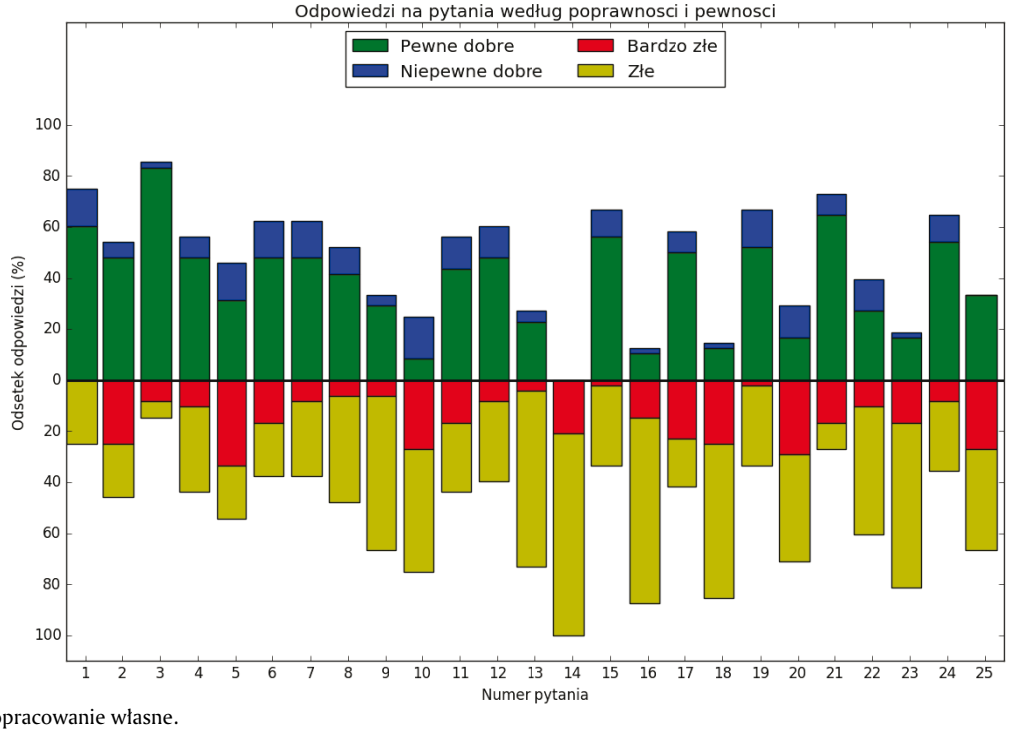
[11] Kod z listingu 3 będzie działał prawidłowo tylko pod warunkiem uprzedniego wykonania kodu z listingów wcześniejszych (tj. 1 i 2). Analogiczne zastrzeżenie dotyczy wszystkich dalszych listingów.

Listing 3. Prezentacja histogramu dla odpowiedzi na poszczególne pytania

```
import matplotlib.pyplot as plt
ind = np.arange(P);   oceny_pytan[:2] = -oceny_pytan[:2]
wpd = plt.bar(ind, oceny_pytan[3], color='g')
wnpd = plt.bar(ind, oceny_pytan[2], color='b', bottom=oceny_pytan[3])
wbz = plt.bar(ind, oceny_pytan[0], color='r')
wz = plt.bar(ind, oceny_pytan[1], color='y', bottom=oceny_pytan[0])
plt.ylabel('Odsetek odpowiedzi (%)');    plt.ylim((-110,140))
plt.title(u'Odpowiedzi na pytania według poprawnosci i pewnosci')
plt.xticks(ind + 0.5, map(str,xrange(1,P+1)))
plt.yticks(xrange(-100,101,20), map(abs,xrange(-100,101,20)))
plt.axhline(color='k', lw=2)
plt.legend((wpd[0], wnpd[0], wbz[0], wz[0]), ('Pewne dobre',
            'Niepewne dobre', u'Bardzo złe', u'Złe'), loc=9,  ncol=2)
plt.show()
```

Źródło: opracowanie własne.

Rysunek 1. Odpowiedzi udzielone na poszczególne pytania testu



Źródło: opracowanie własne.

jak pytania 1 i 21 (te dwa ostatnie w nieco mniejszym stopniu). Spostrzeżenia te uwzględniono przy opracowywaniu pytań testowych w kolejnych latach.

Analiza odpowiedzi na pytania dotyczące poszczególnych wykładów

Pytania zawarte w teście powinny odzwierciedlać możliwie w pełni i równomiernie tematykę wykładów. W ocenie, na ile udało się to osiągnąć, pomocna jest wizualizacja ukazująca pokrycie wykładów pytaniami. Dodatkowo odpowiedzi udzielane na pytania przypisane do danego wykładu pośrednio wskazują poziom opanowania przez studentów wiedzy z obszaru, którego dotyczył wykład. Gdy poziom ten jest niski, stanowi to przesłankę do weryfikacji treści i formy wykładu.

Zamierzone efekty nauczania dotyczą nie tylko wiedzy, ale też opanowania różnego typu efektów kształcenia. Interesujące jest zatem, jak poprawność odpowiedzi prezentowała się po uwzględnieniu rodzaju pytań. W literaturze można spotkać się z różnymi taksonomiami pytań testowych dotyczących programowania komputerów[12]. Tu pytania podzielono

[12] Zob. np. D.I. Chatzopoulou, A.A. Economides, *Adaptive assessment of student's knowledge in* programming courses, „Journal of Computer Assisted Learning" 2010, Vol. 26, No. 4, s. 262, http://dx.doi.org/10.1111/j.1365-2729.2010.00363.x.

na dwa rodzaje, z których każdy obejmował po dwa podrodzaje:

- pytania weryfikujące wiedzę:
  - wymagające wskazania dobrej odpowiedzi wśród złych,
  - wymagające wskazania złej odpowiedzi wśród dobrych;
- pytania weryfikujące umiejętność prawidłowego zastosowania wiedzy:
  - wymagające rozumienia (np. kodu źródłowego fragmentu programu),
  - wymagające (prócz rozumienia kodu) przeprowadzenia prostych obliczeń.

Wszystkie omawiane informacje przedstawiono na pojedynczym diagramie rozproszenia. Odpowiedzi na poszczególne pytania mają na nim formę kółek oznaczonych numerami pytań, o kolorze zależnym od rodzaju pytania. Ich pionowa pozycja zależy od odsetka punktów przyznanych studentom za dane pytanie (spośród wszystkich możliwych do zdobycia za nie), poziomą zaś wyznacza numer wykładu, którego treści ono dotyczyło. Przy każdym kółku umieszczono strzałkę pokazującą, o ile lepiej od przeciętnej na dane pytanie odpowiadali studenci obecni na wykładzie, którego ono dotyczyło (strzałka skierowana w dół oznacza, że odpowiadali oni gorzej). Ponadto kółka oznaczające pytania dotyczące tego samego wykładu starano się rozsunąć w poziomie, tak by zmniejszyć ryzyko nachodzenia ich na siebie – co, wraz z niestandardowym rozmieszczeniem oznaczeń na osi odciętych, dodatkowo skomplikowało kod źródłowy służący do wygenerowania wykresu (patrz listing 4). Rezultat wykonania kodu przedstawiono na rysunku 2.

Patrząc na rysunek 2, trudno dostrzec, by określony rodzaj pytania czy wykład wiązał się ze szczególnie niskim lub wysokim poziomem odpowiedzi. Diagram ujawnia natomiast nierówną liczbę pytań dotyczących poszczególnych wykładów oraz wyraźną przewagę pytań teoretycznych w przypadku wykładów początkowych i pytań praktycznych w przypadku wykładów końcowych. Są to potencjalnie słabe strony testu – wskazują, które elementy wymagają weryfikacji. W przypadku omawianego testu nierównomierność okazała się uzasadniona: treść wykładu – np. definicje i klasyfikacje (wykład 1) albo przykłady kodu (wykład 6) – sprzyjała bowiem formułowaniu pytań jednego rodzaju (dotyczących teorii lub praktyki), a waga poszczególnych wykładów nie była taka sama – np. składnia języka (wykład 2) a kodowanie danych (wykład 5).

Jeśli chodzi o wpływ obecności studentów na poprawność ich odpowiedzi, dla 16 z 25 pytań obecni na dotyczących ich wykładach studenci uzyskali wyniki lepsze od nieobecnych. W przypadku pytań 1 i 5, gdzie różnica na korzyść nieobecnych była wyraźnie widoczna, dokonano weryfikacji formy prezentacji treści, których dotyczyły te pytania, nie dopatrzono się jednak błędów – należy mieć na uwadze, że:

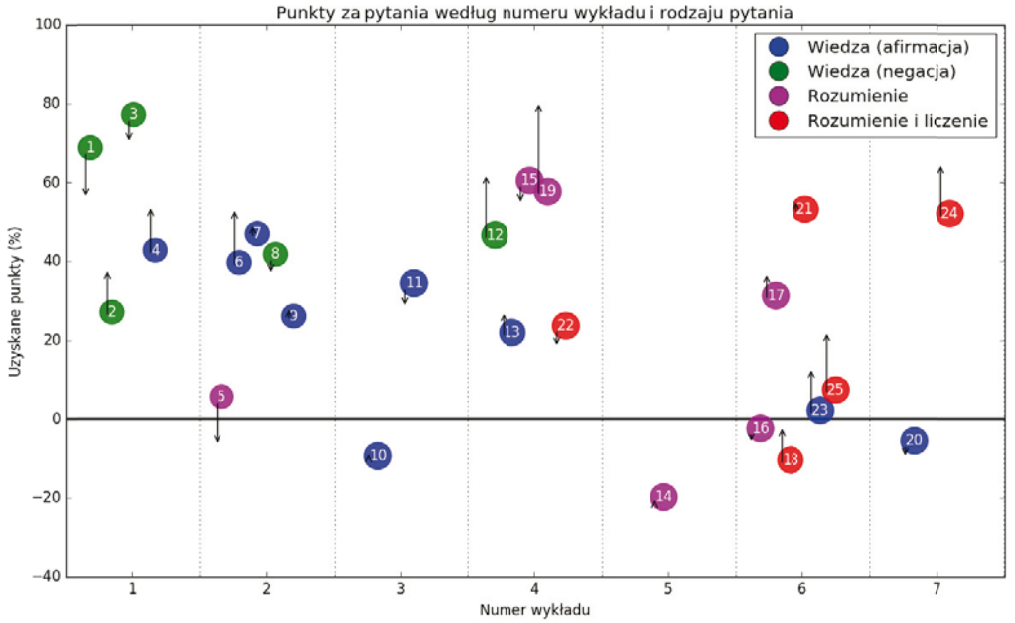
1. test przeprowadzono na zakończenie semestru, a pytania te dotyczyły treści dwóch wykładów z początku semestru,
2. treść wykładów była dostępna dla wszystkich studentów za pośrednictwem platformy e-learningowej,
3. wykłady stanowiły tylko jedną z form nauczania (obok laboratoriów i pracy samodzielnej).

Listing 4. Prezentacja wyników w kontekście zakresu treści i rodzaju pytania

```
sx = np.empty(P);   poz = np.arange(0.5, W+0.5)
przesun = 0.8/(np.histogram(nw, bins=xrange(W+1))[0] + 1.0)
for j,v in enumerate(nw):
    poz[v] += przesun[v];  sx[j] = poz[v]
fig, ax = plt.subplots();   ax.set_xlim([0.5, W+0.5])
plt.axhline(color='k', lw=2);   ax.scatter(sx, sy, s=0)
kolor_pytania={'TN':'g','TP':'b','PR':'m','PO':'r'}
for j in xrange(P):
    ow  = obecnosci.ix[:,numer_wykladu[j]];
    r = np.bincount(ow, weights=oceny[j]) * 100.0
    wy = r.sum() / N;   oy = r[1] / ow.sum();   wx = sx[j]
    ax.annotate(str(j+1), (wx, wy), size=12, color='w',
                bbox=dict(boxstyle="circle", alpha=0.8,
                          color=kolor_pytania[rodzaj_pytania[j]]))
    ax.annotate("", (wx,wy), (wx,oy),
                arrowprops=dict(arrowstyle="<-"))
ax.xaxis.set_ticks(ticks=np.arange(1.5, W+1.5), minor=True)
ax.xaxis.set(ticks=np.arange(1, W+1));  ax.grid(which='minor',axis='x')
plt.xlabel(u'Numer wykładu');    plt.ylabel(u'Uzyskane punkty (%)');
plt.title(u'Punkty za pytania według numeru wykładu i rodzaju pytania')
plt.legend([plt.plot([], k+"o", markersize=16)[0] for k in ("bgmr")],
           ['Wiedza (afirmacja)', u'Wiedza (negacja)', 'Rozumienie',
           'Rozumienie i liczenie'], numpoints=1);
plt.show()
```

Źródło: opracowanie własne.

Rysunek 2. Przyznane punkty według zakresu treści i rodzaju pytania



* Rysunek w wersji kolorowej dostępny jest w internetowym wydaniu czasopisma: www.e-mentor.edu.pl/artykul/index/64/id/1239.

Źródło: opracowanie własne.

Analiza odpowiedzi poszczególnych studentów

Jednym z podstawowych pytań odnoszących się do wyników testu jest pytanie o wzajemne podobieństwo odpowiedzi udzielanych przez różnych studentów. Listing 5 prezentuje kod źródłowy funkcji obliczającej wartość średniego współczynnika podobieństwa Jaccarda[13] pomiędzy odpowiedziami udzielonymi na wszystkie pytania testowe przez dwóch studentów.

W celu przejrzystego rozróżnienia stopni podobieństwa można wykorzystać pseudokolorowanie. W dotychczas omówionych przykładach starano się precyzyjnie osiągnąć zamierzony cel, co wydłużało kod źródłowy. Aby pokazać, że do skutecznej wizualizacji danych wystarczy niewielka liczba poleceń języka Python, w ostatnim przykładzie zrezygnowano z tytułu i opisów osi oraz modyfikacji domyślnych ustawień (poza ustaleniem górnych granic przedziałów). Odpowiedni kod źródłowy zamieszczono na listingu 6.

Listing 5. Obliczanie współczynnika podobieństwa odpowiedzi dwóch studentów

```
def podob(i,j):
    takie_same = 0.0
    for pyt in xrange(P):
        o1 = wyniki.iat[int(i),pyt];   o2 = wyniki.iat[int(j),pyt];
        if o1 == o2:
            takie_same += 1.0
        else:
            s1 = set(o1);   s2 = set(o2)
            takie_same += float(len(s1 & s2))/len(s1 | s2)
    return takie_same / P
```

Źródło: opracowanie własne.

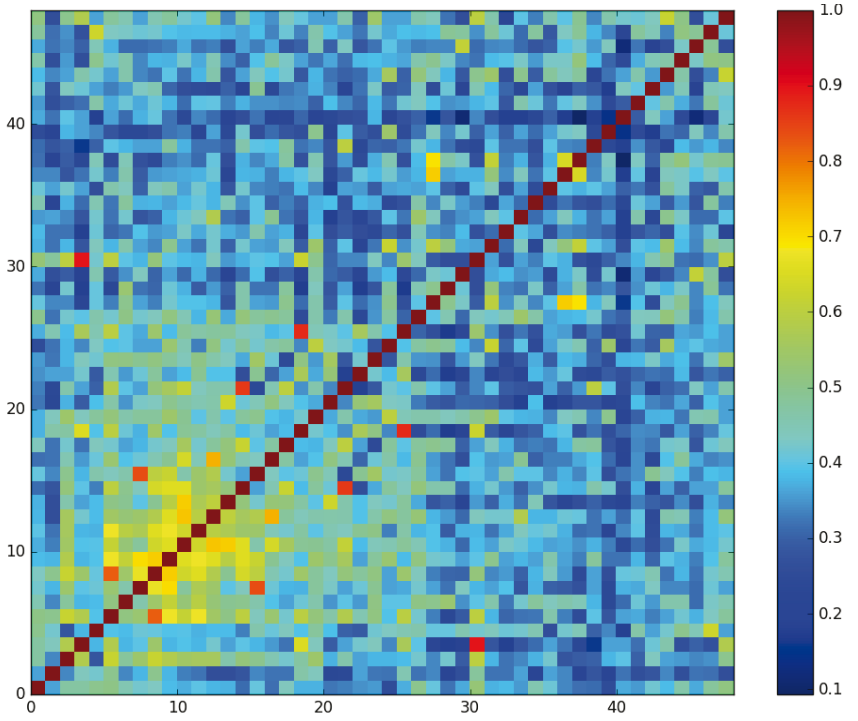
Listing 6. Prezentacja podobieństwa odpowiedzi wszystkich par studentów

```
R = np.fromfunction(np.vectorize(podob), (N, N), dtype=int)
fig, ax = plt.subplots();   ax.set_xlim([0, N]);   ax.set_ylim([0, N])
plt.pcolor(R);   plt.colorbar();   plt.show()
```

Źródło: opracowanie własne.

[13] P. Jaccard, *Étude comparative de la distribution florale dans une portion des Alpes et des Jura*, „Bulletin de la Société Vaudoise des Sciences Naturelles" 1901, Vol. 37, s. 547–579, http://dx.doi.org/10.5169/seals-266450.

Rysunek 3. Stopień podobieństwa odpowiedzi udzielanych przez studentów



* Rysunek w wersji kolorowej dostępny jest w internetowym wydaniu czasopisma: www.e-mentor.edu.pl/artykul/index/64/id/1239.

Źródło: opracowanie własne.

Uzyskany w wyniku jego wykonania wykres (rys. 3) pokazuje, że stopień podobieństwa odpowiedzi udzielanych przez studentów był niski. Należy tu zauważyć, że możliwość wskazania dla każdego pytania więcej niż jednej odpowiedzi powoduje zmniejszenie prawdopodobieństwa przypadkowej zgodności odpowiedzi różnych studentów.

Dzięki jaskrawoczerwonej barwie zwraca uwagę pięć par studentów, które wyróżniają się na tym tle: (4, 31); (6; 9); (8, 16); (15, 22) i (19, 26). Wypada tu wyjaśnić, że choć odczytanie współrzędnych oznaczających numery studentów z rysunku 3 byłoby trudne, to oryginalny wykres wyświetlony z poziomu interpretera języka Python pozwala na ich łatwe odczytanie po przesunięciu wskaźnika myszy do odpowiedniego punktu wykresu. Możliwe jest też wtedy m.in. powiększanie wykresu i przesuwanie jego widocznego fragmentu, a także zapisanie go w jednym z popularnych formatów grafiki rastrowej lub wektorowej.

Przypadek oczywisty stanowi sytuacja, w której różni studenci odpowiadali pewnie i perfekcyjnie (a zatem musieli też zakreślać te same – dobre odpowiedzi). Przypadek ten nie dotyczył jednak żadnej z wymienionych pięciu par studentów, dlatego przyczyn zgodności ich odpowiedzi należałoby się doszukiwać albo we wspólnym uczeniu się z tych samych materiałów (np. znalezionych w internecie), które zawierały braki lub błędy merytoryczne, lub – co niestety bardziej prawdopodobne – „ściąganiu" studentów od siebie (do weryfikacji tego przydatna byłaby informacja o miejscach, które studenci zajmowali podczas testu).

Zalety wykorzystania oprogramowania Anaconda na tle Microsoft Excel

Analiza graficzna danych edukacyjnych nie musi być prowadzona w oparciu o język Python, z wykorzystaniem oprogramowania Anaconda lub analogicznego. Naturalne rozwiązanie alternatywne stanowi wykorzystanie popularnego programu Microsoft Excel. Poniżej wskazane zostaną przesłanki przemawiające za wyborem Anacondy.

Podstawowa różnica między tymi dwoma środowiskami polega na domyślnym sposobie dochodzenia do pożądanego celu: poprzez jego wyspecyfikowanie w postaci programu w języku Python lub użycie interaktywnego kreatora wykresów. Choć Anaconda również dysponuje analogicznym narzędziem interaktywnym (poprzez bibliotekę Pivottablejs[14]), warto mieć świadomość zalet posługiwania się kodem źródłowym. Za trzy najważniejsze z nich należy uznać:

- zakres adaptacji formy wizualizacji danych, dalece wykraczający poza możliwości narzędzia interaktywnego;

[14] N. Kruchten, *PivotTable.js integration for Jupyter/IPython Notebook Web-based Python Data Analysis*, https://pypi.python.org/pypi/pivottablejs, [17.04.2016].

- szybkość wprowadzania zmian: dzięki pętlom i instrukcjom warunkowym możliwe jest zapisanie w kilku wierszach kodu źródłowego reguł pozwalających uniknąć nawet kilkudziesięciu minut mozolnego wybierania i modyfikowania ustawień wizualizacji drobnych elementów wykresu;
- łatwość ponownego wykorzystania: nawet gdy rozszerza się zakres danych wejściowych czy dodaje do wykresu kolejną zmienną, wymaga to jedynie niewielkiej modyfikacji kodu; w przypadku narzędzia interaktywnego istotna zmiana treści często oznacza konieczność ponownego ręcznego dostosowania szczegółowych ustawień wizualizacji.

Jako że program Microsoft Excel również posiada wbudowany interpreter języka programowania VBA (*Visual Basic for Applications*), który można traktować jako rozwiązanie alternatywne dla interpretera języka Python dostępnego w oprogramowaniu Anaconda, dokonano porównania tych dwóch rozwiązań w oparciu o dziewięć kluczowych kryteriów użytkowych. Jego rezultaty, zamieszczone w tabeli 1, dobitnie pokazują wyższość rozwiązania opartego na języku Python.

Tabela 1. Anaconda/Python a Excel/VBA – porównanie

Kryterium	Anaconda/Python	Excel/VBA
Koszt	Bezpłatny	Płatny
Perspektywy rozwoju	Stabilne	Niejasne (zastępowanie VBA przez JavaScript)
Składnia języka	Przejrzysta, zwięzła	Skomplikowana, rozwlekła
Społeczność online	Bardzo duża, rosnąca (TIOBE rank 5)	Duża, malejąca (TIOBE rank 14)
Centralny katalog rozszerzeń	Jest (PyPI: Python Package Index)	Brak
Modyfikacja formy wizualizacji	Praktycznie nieograniczona	Wąska
Praca w chmurze	Tak	Nie
Sterowanie programem Excel	Tak (poprzez bibliotekę xlwings)	Tak
Osadzanie w notatniku Jupyter	Tak	Nie

Źródło: opracowanie własne z wykorzystaniem *TIOBE Index for April 2016*, http://www.tiobe.com/tiobe_index, [17.04.2016].

Podsumowanie

Opisane studium przypadku pokazuje, że graficzna analiza danych może być z powodzeniem wykorzystana do analizy wyników testów i dostarczyć wielu informacji użytecznych w kontekście procesu doskonalenia metod i środków nauczania oraz weryfikacji jego wyników. Jednocześnie zademonstrowano użyteczność do tego celu interpretera języka Python. Język ten, dzięki dostępności bibliotek obliczeniowych (np. NumPy) i wizualizacyjnych (np. Matplotlib), stwarza szerokie możliwości prowadzenia analizy danych edukacyjnych. Szczególną wygodę korzystania z niego zapewnia oprogramowanie Anaconda, obejmujące, prócz samego interpretera i pakietu bibliotek, notatnik Jupyter oraz narzędzie zarządzania modułami Conda. Przeprowadzone porównanie ujawnia liczne przewagi tego środowiska nad programem Microsoft Excel.

Bibliografia

Chatzopoulou D.I., Economides A.A., *Adaptive assessment of student's knowledge in programming courses*, „Journal of Computer Assisted Learning" 2010, Vol. 26, No. 4, s. 258–269, http://dx.doi.org/10.1111/j.1365-2729.2010.00363.x.

Eppler J.M., Helias M., Muller E., Diesmann M., Gewaltig M.-O., *PyNEST: A Convenient Interface to the NEST Simulator*, „Frontiers in Neuroinformatics" 2008, No. 2, art. 12, http://dx.doi.org/10.3389/neuro.11.012.2008.

Goggins S., Xing W., Chen X., Chen B., Wadholm B., *Learning Analytics at „Small" Scale: Exploring a Complexity-Grounded Model for Assessment Automation*, „Journal of Universal Computer Science" 2015, Vol. 21, No. 1, s. 66–92, http://www.jucs.org/jucs_21_1/learning_analytics_at_small.

Gorelick M., Ozsvald I., *High Performance Python: Practical Performant Programming for Humans*, O'Reilly Media, Sebastopol 2014.

Gross A., *A brief history of education's big data debate*, „Education Dive", 07.05.2014, http://www.educationdive.com/news/a-brief-history-of-educations-big-data-debate/258602.

Jaccard P., *Étude comparative de la distribution florale dans une portion des Alpes et des Jura*, „Bulletin de la Société Vaudoise des Sciences Naturelles" 1901, Vol. 37, s. 547–579, http://dx.doi.org/10.5169/seals-266450.

Kitchin R., Lauriault T.P., *Small data in the era of big data*, „GeoJournal" 2015, Vol. 80, No. 4, s. 463–475, http://dx.doi.org/10.1007/s10708-014-9601-7.

Kruchten N., *PivotTable.js integration for Jupyter/IPython Notebook Web-based Python Data Analysis*, https://pypi.python.org/pypi/pivottablejs.

Miciuła I., Miciuła K., *Kluczowe trendy dla budowania biznesu w branży big data*, „Zeszyty Naukowe Uniwersytetu Szczecińskiego. Studia Informatica" 2015, nr 36(863), s. 51–63, http://dx.doi.org/10.18276/si.2015.36-04.

Papamitsiou Z., Economides E., *Learning Analytics and Educational Data Mining in Practice: A Systematic Literature Review of Empirical Evidence*, „Educational Technology & Society" 2014, Vol. 17, No. 4, s. 49–64, http://www.jstor.org/stable/jeductechsoci.17.4.49.

Płoszajski P., *Big Data: nowe źródło przewag i wzrostu firm*, „e-mentor" 2013, nr 3(50), s. 5–10, http://www.e-mentor.edu.pl/artykul/index/numer/50/id/1016.

Rossant C., *Learn IPython for interactive Python programming, high-performance numerical computing, and data visualization*, Packt Publishing, Birmingham 2013.

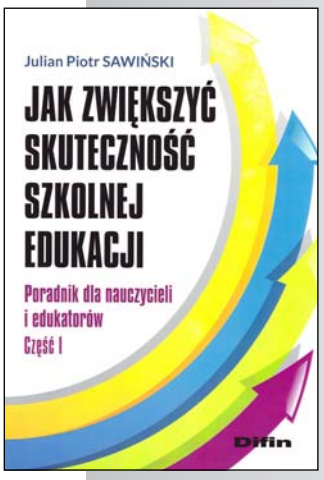
TIOBE Index for April 2016, http://www.tiobe.com/tiobe_index.

Graphical analysis of educational data using Python

In practice, the high quality of educational processes and content can hardly be achieved and maintained without monitoring. And since tracking the results is not enough, it is necessary to find existing flaws and identify their causes. In this paper, it is argued that even small data sets such as students' test results can provide valuable information useful to improve further iterations of teaching and learning outcomes verification processes and educational materials. Next, the appropriateness of graphical analysis for this purpose is pointed out taking its simplicity even for non-statisticians and its ability to visualize entire small data sets with high precision into account. However, the primary aim of this paper is to provide practical examples showing that the Python programming language (with a selection of specialized modules) can be used in a convenient and effective manner for graphical analysis of small educational data sets. For the purpose of demonstration case study, actual test results were used. Analysis examined: correctness and confidence of students answering respective test questions, correlations between results based on question type and relevant content area, and also similarities of answers given by different students. Also, a Python-based software environment for graphical analysis has been compared to Microsoft Excel.

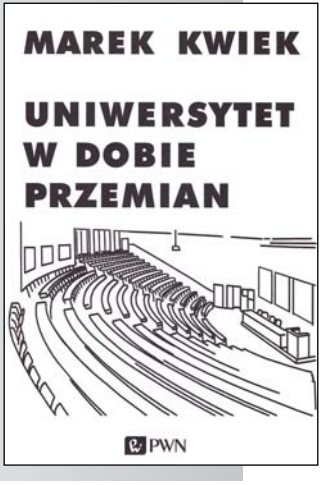
Autor jest doktorem nauk technicznych w zakresie informatyki oraz doktorem habilitowanym nauk ekonomicznych w zakresie nauk o zarządzaniu. Obecnie kieruje Zakładem Inżynierii Oprogramowania Instytutu Informatyki w Zarządzaniu Uniwersytetu Szczecińskiego. Od ponad 10 lat zaangażowany jest w popularyzowanie w Polsce języka Python. Jest autorem publikacji, które należą do pierwszych na ten temat w języku polskim: podręcznika oraz internetowego kursu języka Python. Wśród jego zainteresowań badawczych od lat znajduje się problematyka nauczania programowania i jego informatycznego wspomagania.

POLECAMY



Julian Piotr Sawiński
Jak zwiększyć skuteczność szkolnej edukacji. Poradnik dla nauczycieli i edukatorów. Część 1 i 2
Difin, Warszawa 2015 i 2016

Obie części niniejszej publikacji stanowią całość i są w odpowiedzią na pytanie stawiane od wielu lat przez osoby działające w sektorze szkolnictwa: jak zwiększyć skuteczność edukacji szkolnej? Ten poradnik nie tylko prezentuje jak najbardziej kluczowe zagadnienia, które wpływają na jakość kształcenia, ale jest też wzbogacony o zadania do samodzielnej pracy i porady dla nauczycieli. Łącznie w 24 rozdziałach poruszane są takie zagadnienia jak: innowacyjność w nauczaniu, motywowanie uczniów, neurodydaktyka, kreatywność, zastosowanie filmów czy kluczowe kompetencje ucznia i nauczyciela. Zapraszamy do zapoznania się z opisywanymi publikacjami nie tylko grono pedagogiczne i studentów pedagogiki, ale również wszystkich, którzy mają wpływ na przyszłość edukacji szkolnej w Polsce.
Publikację można nabyć w księgarni internetowej wydawnictwa:
http://www.ksiegarnia.difin.pl/



Marek Kwiek
Uniwersytet w dobie przemian. Instytucje i kadra akademicka w warunkach rosnącej konkurencji
Wydawnictwo Naukowe PWN, Warszawa 2016

Prezentowana publikacja to wynik wieloletnich badań autora prowadzonych zarówno na polskich, jak i na europejskich uczelniach. Została podzielona na dwie części: pierwsza (teoretyczna) oparta jest na pytaniach badawczych, które zdaniem autora są kluczowe dla analizy szkolnictwa wyższego w Polsce. Część druga zawiera natomiast analizę tematu samej kadry akademickiej, której podstawą były badania przeprowadzone na grupie ponad 17 tys. reprezentantów uczelni europejskich. Wśród zagadnień poruszanych przez autora znalazły się takie kwestie, jak: umiędzynarodowienie badań, kolegialność akademicka, różnice międzypokoleniowe wśród kadry dydaktycznej, a także szanse rozwojowe i kariera kadry akademickiej.
Publikację można nabyć w księgarni internetowej wydawnictwa: **http://ksiegarnia.pwn.pl**