Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Jakub Węgłowski
Nr albumu: 430620

Struktura preferencji w budżetach obywatelskich

Praca licencjacka
na kierunku MATEMATYKA

Praca wykonana pod kierunkiem
dr. hab. Piotra Skowrona
Instytut Informatyki

Warszawa, Czerwiec 2023

Streszczenie

W niniejszej pracy zbadaliśmy, czy wyborcy w głosowaniach na budżet obywatelski chętniej głosują na podobne do siebie projekty. Zdefiniowaliśmy dwa sposoby mierzenia dystansu między projektami – jeden wynikający z ich nazw, zaś drugi wynikający z oddanych głosów. Na podstawie analizy rzeczywistych instancji wyborczych zbadaliśmy korelację pomiędzy nimi. Dla poszczególnych instancji przeprowadziliśmy klastrowanie zbioru projektów względem obu miar odległości. Statystyczna analiza wykazała, że w niektórych instancjach wyborczych głosujący chętniej popierają podobne do siebie projekty.

Słowa kluczowe

Budżet obywatelski, Budżet partycypacyjny, Preferencje wyborcze, Approval voting, Klastrowanie

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.1 Matematyka

Klasyfikacja tematyczna

62H20 - Measures of association
91B14 - Social choice
91C20 - Clustering in the social and behavioral sciences

Tytuł pracy w języku angielskim

Preference structure in participatory budgeting

Spis treści

Wprowadzenie

Niniejsza praca dotyczy struktury preferencji wyborców w głosowaniach typu "approval voting". Jest to typ głosowania, w którym wyborcy głosują na kandydatów, wskazując tych, których akceptują. Jest to prosta metoda, gdyż nie wymaga od głosującego na przykład szeregowania kandydatów w kolejności od najbardziej do najmniej popieranego, bądź przypisywania kandydatom wartości punktowych. Głosujący wybiera jedynie tych kandydatów, których jest skłonny zaakceptować. Celem procesu wyborczego jest końcowe wyłonienie zwycięskiego komitetu, złożonego z pewnego podzbioru zbioru kandydatów [8].

Metoda ta nie sprawdza się zatem w przypadku wyborów, w których ostateczny zwycięzca musi być tylko jeden, jak w przypadku wyborów prezydenckich, na burmistrza miasta itp. Może ona jednakże sprawdzić się wszędzie tam, gdzie ostatecznie wybranych może zostać wielu spośród kandydujących. Myśląc o kandydatach, często mamy na myśli osoby fizyczne – potencjalnych posłów, senatorów bądź radnych. Okazuje się jednak, że ten proces wyborczy można z powodzeniem zaaplikować również do wyborów, w których kandydatami nie są ludzie.

W wielu miastach w Polsce od lat mieszkańcy co roku głosują na projekty, których realizacja jest finansowana przez władze miasta w ramach tzw. budżetu obywatelskiego. Idea jest prosta: mieszkańcy zgłaszają projekty, które często dotyczą spraw bezpośrednio wpływających na ich życie codzienne. Następnie w głosowaniu wybierane są te projekty, które mają zostać zrealizowane. Miasto wygospodarowuje część środków z budżetu, które zostaną poświęcone na finansowanie zwycięskich projektów. W tej sytuacji wyborcami są mieszkańcy danego miasta bądź jego dzielnicy, zaś kandydatami są projekty, które zostały zgłoszone do głosowania.

Zasadniczym celem niniejszej pracy jest zaprezentowanie metody, przy pomocy której możliwa będzie odpowiedź na pytanie, czy wyborcy chętniej głosują na podobne do siebie projekty. W przypadku wyborów dotyczących budżetu obywatelskiego, wyniki takiej analizy mogą zostać poddane skutecznej weryfikacji, gdyż każdy ze zgłoszonych projektów ma przypisaną sobie nazwę bądź opis, które jasno wskazują na obszar życia, którego dany projekt dotyczy. W przypadku projektów zgłoszonych w głosowaniu nad budżetem obywatelskim łatwo jest więc stwierdzić na podstawie nazw i opisów, czy i jak bardzo dane dwa projekty są do siebie podobne.

W pierwszej kolejności przeanalizujemy zbiory zgłoszonych projektów jedynie na podstawie wyników głosowań. W tym celu określimy na zbiorze projektów odpowiednią metrykę. Następnie, przy pomocy modelu lingwistycznego, zbadamy podobieństwo projektów na podstawie ich nazw, zgłoszonych w procesie zbierania propozycji poddanych pod głosowanie. Dalej, zbadamy korelację pomiędzy dystansem lingwistycznym, a dystansem wynikającym z głosowania. Na zakończenie, zaprezentujemy metodę poszukiwania rozbicia zbioru projektów na klastry zawierające projekty do siebie podobne.

Rozdział 1

Podstawowe pojęcia

Niech $V = \{1, ..., N\}$ oznacza zbiór wyborców, zaś $C = \{c_1, ..., c_m\}$ – zbiór kandydatów (projektów). Wyborami będziemy nazywać parę uporządkowaną $E = (V, C)$. Zgodnie z wyjaśnieniem zawartym we wstępie, głosowanie odbywa się poprzez wskazanie kandydatów, których dany wyborca akceptuje. Dla każdego z kandydatów $c \in C$ niech:

$$S(c) = \{i \in V : \text{ wyborca } i \text{ akceptuje } c\}$$

Oczywiście, wprost z definicji mamy $S(c) \subseteq V$.

Istnieje także potrzeba wprowadzenia pewnej miary odległości pomiędzy zbiorami, aby móc sformalizować metody badania podobieństwa pomiędzy wyborcami lub projektami. W literaturze funkcjonuje wiele różnych sposobów definiowana odległości pomiędzy zbiorami dyskretnymi, w tej pracy będziemy szczególnie skupiać się na dwu z nich:

Definicja 1 *Niech $A, B \neq \emptyset$ będą zbiorami skończonymi, tj. $|A|, |B| < \infty$. Odległością Hamminga pomiędzy zbiorami A i B nazywamy liczbę elementów ich różnicy symetrycznej:*

$$H(A, B) = |A \triangle B|$$

Odległość Hamminga nie uwzględnia w żaden sposób liczebności zbiorów, których wzajemną odległość mierzy. Prowadzi to do sytuacji sprzecznych z naszą intuicją i oczekiwaniami – zobaczmy następujący przykład:

Przykład 1 *Niech $A = \{1, 2, 3\}$ oraz $B = \{3, 4, 5\}$. Wówczas $A \triangle B = \{1, 2, 4, 5\}$, zatem $H(A, B) = 4$. Niech teraz $C = \{1, 2, ..., 100\}$ oraz $D = \{3, 4, ..., 102\}$. Również mamy $H(C, D) = |\{1, 2, 101, 102\}| = 4$. Intuicyjnie można jednak oczekiwać, że zbiory C i D są do siebie bardziej podobne, niż zbiory A i B, gdyż względem liczby wszystkich swoich elementów mają znacznie liczniejsze przecięcie. Nieznacznie formalizując tę intuicję, możemy zauważyć, że:*

$$\frac{|A \cap B|}{|A \cup B|} = \frac{1}{5} \text{ oraz } \frac{|C \cap D|}{|C \cup D|} = \frac{98}{102}$$

Powyższy przykład ilustruje potrzebę uwzględnienia liczby elementów zbiorów, których odległość mierzymy. Z tej obserwacji wynika poniższa definicja:

Definicja 2 *Niech $A, B \neq \emptyset$ będą zbiorami skończonymi, tj. $|A|, |B| < \infty$. Odległością Jaccarda pomiędzy zbiorami A i B nazywamy liczbę elementów ich różnicy symetrycznej, podzieloną przez liczbę elementów ich sumy:*

$$J(A, B) = \frac{|A \triangle B|}{|A \cup B|} = \frac{H(A, B)}{|A \cup B|}$$

Jak łatwo zauważyć, definicja odległości Jaccarda jest powiązana z rachunkiem zawartym w Przykładzie 1:

$$J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} = 1 - \frac{|A \cap B|}{|A \cup B|}$$

Stosunek liczności przecięcia do liczności sumy jest intuicyjnie miarą wzajemnego podobieństwa zbiorów. Liczbę $\frac{|A \cap B|}{|A \cup B|}$ w literaturze nazywa się indeksem Jaccarda. Gdy zbiory są podobne, odległość między nimi powinna być niewielka. Rzeczywiście, gdy indeks Jaccarda dla zbiorów A, B jest bliski 1, odległość $J(A,B)$ jest bliska 0.

Przykład 2 *Dla zbiorów z Przykładu 1 mamy:*

$$J(A,B) = 1 - \frac{1}{5} = \frac{4}{5} \text{ oraz } J(C,D) = 1 - \frac{98}{102} = \frac{4}{102}$$

Ten rezultat rzeczywiście wyraża pewną intuicję odnośnie tego, którą z tych par tworzą zbiory bardziej do siebie podobne.

Twierdzenie 1 *Odległość Jaccarda spełnia aksjomaty metryki, tj.:*

1. *dla dowolnych niepustych, skończonych zbiorów A, B zachodzi $J(A,B) \geq 0$ oraz $J(A,B) = 0 \Leftrightarrow A = B$.*
2. *dla dowolnych niepustych, skończonych zbiorów A, B zachodzi $J(A,B) = J(B,A)$.*
3. *dla dowolnych niepustych, skończonych zbiorów A, B, C zachodzi $J(A,B) \leq J(A,C) + J(C,B)$.*

Dowód.

1. *Dodatnia określoność.* Niech A, B będą skończone oraz niech przynajmniej jeden z nich będzie niepusty. Funkcja $|.|$, licząca elementy zbiorów skończonych, przyjmuje wartości naturalne, a więc nieujemne. Stąd oczywiście:

   $$J(A,B) = \frac{|A \triangle B|}{|A \cup B|} \geq 0$$

   Ponadto, gdy $A = B$, to $A \cup B = A \cap B$, czyli $A \triangle B = \emptyset$, zatem $|A \triangle B| = 0$, czyli $J(A,B) = 0$. W drugą stronę, jeśli $J(A,B) = 0$, to $|A \triangle B| = 0$. Musi być zatem $A \triangle B = \emptyset$, czyli $(A \setminus B) \cup (B \setminus A) = \emptyset$. Stąd wynika, że $A \setminus B = B \setminus A = \emptyset$, czyli rzeczywiście $A = B$.

2. *Symetryczność.* Własność symetrii jest oczywista i wynika natychmiastowo z faktu, że różnica symetryczna zbiorów oraz suma zbiorów to działania przemienne:

   $$A \triangle B = B \triangle A \text{ oraz } A \cup B = B \cup A$$

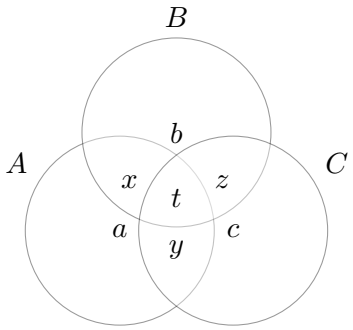
3. *Nierówność trójkąta.* [3] Przypuśćmy przeciwnie, że istnieją niepuste zbiory A, B, C, dla których zachodzi nierówność:

   $$\frac{|A \triangle B|}{|A \cup B|} > \frac{|A \triangle C|}{|A \cup C|} + \frac{|B \triangle C|}{|B \cup C|}$$

   Powyższą nierówność równoważnie można zapisać jako:

   $$\begin{aligned} 1 - \frac{|A \cap B|}{|A \cup B|} &> 1 - \frac{|A \cap C|}{|A \cup C|} + 1 - \frac{|B \cap C|}{|B \cup C|} \\ \frac{|A \cap B|}{|A \cup B|} &< \frac{|A \cap C|}{|A \cup C|} + \frac{|B \cap C|}{|B \cup C|} - 1 \end{aligned}$$

Wprowadźmy oznaczenia jak na poniższym rysunku, literami oznaczając liczbę elementów poszczególnych podzbiorów zbioru $A \cup B \cup C$:



Rysunek 1.1: Diagram Venna dla zbiorów A, B, C w położeniu ogólnym.

Wówczas ostatnia nierówność przybiera postać:

$$\frac{x+t}{a+b+x+y+z+t} < \frac{y+t}{a+c+x+y+z+t} + \frac{z+t}{b+c+x+y+z+t} - 1 \tag{1.1}$$

W celu uproszczenia zapisu, oznaczmy $S := |A \cup B \cup C| = a+b+c+x+y+z+t$, wtedy nierówność (1.1) przyjmuje postać:

$$\frac{x+t}{S-c} < \frac{y+t}{S-b} + \frac{z+t}{S-a} - 1 \tag{1.2}$$

Dalej zauważamy, że dla $m, n \geq 0$ takich, że $n > m$ oraz dowolnego $k \geq 0$ mamy $\frac{m}{n} \leq \frac{m+k}{n+k}$, ponieważ nierówność ta jest równoważna nierówności $mn + mk \leq mn + nk$, która z kolei jest równoważna nierówności $0 \leq k(n-m)$ – ta zaś jest oczywista biorąc pod uwagę założenia. Możemy zatem napisać, że:

$$\frac{y+t}{S-b} \leq \frac{y+t+b}{S}$$
$$\frac{z+t}{S-a} \leq \frac{z+t+a}{S}$$

Oczywiście mamy także:

$$\frac{x+t}{S} \leq \frac{x+t}{S-c}$$

Wobec powyższych obserwacji nierówność (1.2) implikuje:

$$\frac{x+t}{S} \leq \frac{x+t}{S-c} < \frac{y+t}{S-b} + \frac{z+t}{S-a} - 1 \leq \frac{y+t+b}{S} + \frac{z+t+a}{S} - 1$$

Co z kolei prowadzi do wniosku, że:

$$x + t < (b + y + t) + (a + z + t) - S$$

Z powyższego natychmiast wynika, że:

$$\begin{aligned} x+t &< -c-x+t \\ 2x &< -c \end{aligned}$$

Co prowadzi do sprzeczności, gdyż lewa strona ostatniej nierówności jest nieujemna, zaś lewa jest niedodatnia, a nierówność jest ostra. Ta obserwacja kończy już dowód całego twierdzenia – wykazaliśmy, że dystans Jaccarda spełnia aksjomaty metryki.

■

Stwierdzenie 1 *Dla dowolnych niepustych, skończonych zbiorów A, B zachodzi*

$$0 \leq J(A,B) \leq 1$$

Ponadto $J(A,B) = 1 \Leftrightarrow A \cap B = \emptyset$.

Dowód.

Nierówność $J(A,B) \geq 0$ pokazaliśmy powyżej. Dla pokazania nierówności $J(A,B) \leq 1$ wystarczy zauważyć, że dla dowolnych zbiorów A, B mamy $A \triangle B \subseteq A \cup B$, stąd (jeśli przynajmniej jeden ze zbiorów jest niepusty) zauważamy:

$$\begin{aligned} |A \triangle B| &\leq |A \cup B| \\ J(A,B) = \frac{|A \triangle B|}{|A \cup B|} &\leq 1 \end{aligned}$$

Dalej, jeśli $A \cap B = \emptyset$ to $A \triangle B = A \cup B$, zatem $J(A,B) = 1$. W drugą stronę, jeśli $J(A,B) = 1$, to:

$$\begin{aligned} |A \triangle B| &= |A \cup B| \\ |(A \cup B) \setminus (A \cap B)| &= |A \cup B| \\ |A \cup B| - |A \cap B| &= |A \cup B| \\ |A \cap B| &= 0 \\ A \cap B &= \emptyset \end{aligned}$$

Tym samym dowód stwierdzenia został zakończony.

■

Ostatnim pojęciem, jakie zdefiniujemy, będzie rozbicie zbioru.

Definicja 3 *Rozbiciem zbioru A nazywamy każdy układ zbiorów $A_1, ..., A_n$ taki, że:*

1. *zbiory te są parami rozłączne, tj. dla $i \neq j$ mamy $A_i \cap A_j = \emptyset$*
2. *suma tych zbiorów daje zbiór A, tj. $\bigcup_{i=1}^{n} A_i = A$.*

Rozdział 2

Analiza preferencji wyborców

2.1. Źródło danych – Pabulib

Do analizy zawartej w tej pracy wykorzystaliśmy dane pochodzące z rzeczywistych głosowań nad budżetem obywatelskim, pochodzące ze strony internetowej działającej pod patronatem Uniwerystetu Jagiellońskiego `http://pabulib.org/`. Jest to baza danych, w której umieszczono pliki zawierające kompleksową informację na temat procesów wyborczych. Pliki umieszczone na stronie zawierają wszystkie niezbędne informacje ogólne dotyczące danego głosowania, w szczególności pełną listę zgłoszonych projektów wraz z nazwami, a także pełną listę głosujących wraz z numerami projektów, które każdy z wyborców akceptuje. W celu dokładnego poznania zawartości plików zawartych w bazie danych odsyłamy do pliku szeroko omawiającego tę kwestię [1].

2.2. Model lingwistyczny

W celu znalezienia odpowiedzi na postawione we wstępnie pytanie badawcze: "Czy wyborcy częściej głosują na podobne projekty?", należy sformalizować poszczególne jego składowe. Na początku należy rozwiązać kwestię określenia, jak bardzo dwa projekty są do siebie podobne. W tym celu użyjemy modelu lingwistycznego opartego na metodach sztucznej inteligencji, do którego podłączona została wytrenowana wcześniej baza słów w języku polskim [2]. Każdemu słowu przyporządkowany jest wektor z przestrzeni $(\mathbb{R}_+)^{300}$, dzięki czemu dysponujemy informacjami na temat wzajemnego podobieństwa słów umieszczonych w bazie danych. W użytym modelu podobieństwo między dwoma słowami oblicza się jako cosinus kąta pomiędzy odpowiadającymi im wektorami. Wynika stąd, że długość wektorów słów nie ma znaczenia, gdyż cosinus kąta zależy jedynie od ich kierunku – całość informacji niesionej przez wektor pojedynczego słowa jest zawarta w kierunku, jaki ten wektor wyznacza.

Następnie wystarczy zauważyć, że zdanie w języku złożone jest ze słów, a zatem wektor zdania powinien być sumą wektorów poszczególnych słów. Aby nie wyróżniać niepotrzebnie jednych słów względem drugich (tzn. tych, których wektory mają większą bądź mniejszą normę euklidesową), każdy z wektorów słów zostanie unormowany. Operacja ta nie powoduje utraty informacji, gdyż zgodnie z uwagą zawartą w poprzednim akapicie całość informacji jest zawarta w kierunku wektora. Wprowadzając oznaczenia, dane jest zdanie s oraz składające się na nie słowa $w_1, ..., w_n$. Każdemu ze słów w_i w bazie danych odpowiada wektor $v(w_i) \in (\mathbb{R}_+)^{300}$. Na początku niech:

$$\tilde{v}(s) = \sum_{i=1}^{n} \frac{v(w_i)}{\|v(w_i)\|}$$

Następnie, wektor $v(s)$ zdania s jest określony jako unormowany wektor $\tilde{v}(s)$:

$$v(s) = \frac{\tilde{v}(s)}{\|\tilde{v}(s)\|}$$

Okazuje się, że w takiej sytuacji można nadać interpretację dobrze znanej z wykładów algebry liniowej wartości cosinusa kąta między niezerowymi wektorami:

$$\cos\left(\angle(v(s_i), v(s_j)\right) = \frac{\langle v(s_i), v(s_j)\rangle}{\|v(s_i)\| \cdot \|v(s_j)\|} = \langle v(s_i), v(s_j)\rangle$$

Gdzie oczywiście ostatnia równość zachodzi dlatego, że wektory te mają długość 1. Z samej konstrukcji modelu wynika, że cosinus kąta między wektorami zdań jest miarą ich wzajemnego podobieństwa. Warto dodać, że ponieważ wszystkie współrzędne wektorów są liczbami dodatnimi, wartości cosinusów będą należały jedynie do przedziału $[0, 1]$. Geometrycznie oznacza to, że im bardziej wektory zdań są bliskie bycia prostopadłymi, tym mniej podobne do siebie będą te zdania. Odwrotnie zaś, im bardziej równoległe będą dwa wektory zdań, tym bardziej podobne będą te zdania.

W celu określenia podobieństwa projektów zgłoszonych w wyborach zaadaptujemy opisaną powyżej metodę. Jako zdania s_i wykorzystamy nazwy (opisy) projektów, które zostały zaproponowane podczas rejestracji w procesie wyborczym. W ten sposób każdy z projektów $c_i \in C$ możemy utożsamić z jego nazwą, która jest zdaniem w języku polskim: s_i. Teraz, w celu określenia odległości[1] pomiędzy dwoma projektami, wystarczy położyć:

$$d_L\left(c_i, c_j\right) := 1 - \cos\left(\angle(v(s_i), v(s_j)\right)$$

Na początek przedstawimy działanie tego algorytmu[2] na prostym przykładzie – zbadamy wzajemne podobieństwo wybranych trzech nazw projektów zgłoszonych w wyborach na warszawskim Ursynowie w roku 2019:

1. "Szkolne prace domowe - mniej, ciekawiej, inaczej"
2. "Prawo mówi wyraźnie, że prace domowe są tylko dobrowolne! Plakaty informujące o dobrowolności prac domowych"
3. "Akademia starej motoryzacji cykl wykładów dotyczący motoryzacji i zlot zabytkowych samochodów na Ursynowie"

Naturalnie należy się spodziewać (de facto, w kontekście całej analizy, należy tego oczekiwać), że pierwsze projekty dwa wykażą większe podobieństwo między sobą, niż względem trzeciego projektu. Spójrzmy, co zwraca kod:

[1] Indeks dolny L pochodzi od "lingwistyczny"

[2] Całość kodu wykorzystanego w pracy dostępna jest na Githubie: `https://github.com/jakubweglowski/computational-social-choice`. Do oceny podobieństwa zaadaptowany został kod dostępny na stronie: `https://github.com/ashokc/Bow-to-Bert/blob/master/fasttext_sentence_similarity.py`

```
Cosine Similarity:
 Prawo mówi wyraźnie, że prace domowe są tylko dobrowolne! Plakaty informujące o dobrowolności prac domowych
&
 Szkolne prace domowe - mniej, ciekawiej, inaczej : 0.7018704

Cosine Similarity:
 Akademia starej motoryzacji cykl wykładów dotyczący motoryzacji i zlot zabytkowych samochodów na Ursynowie
&
 Szkolne prace domowe - mniej, ciekawiej, inaczej : 0.39341688

Cosine Similarity:
 Akademia starej motoryzacji cykl wykładów dotyczący motoryzacji i zlot zabytkowych samochodów na Ursynowie
&
 Prawo mówi wyraźnie, że prace domowe są tylko dobrowolne! Plakaty informujące o dobrowolności prac domowych : 0.5326125
```

Rysunek 2.1: Ocena podobieństwa pomiędzy przykładowymi trzema nazwami projektów.

Rzeczywiście, efekt działania kodu jest zgodny z naszymi oczekiwaniami. Warto w tym miejscu zaznaczyć, że konkretne wartości podobieństwa, które zwraca algorytm, nie grają istotnej roli. Zamiast tego, wyniki należy interpretować na podstawie ich uporządkowania monotonicznego. Posługując się tym rozumowaniem, powyższy wynik działania kodu należy interpretować następująco: skoro $0.702 > 0.533 > 0.393$, to zdania 1. i 2. są do siebie bardziej podobne, niż zdania 2. i 3. oraz 1. i 3. Z kolei zdania 2. i 3. są bardziej podobne, niż 1. i 3. Nie należy zaś interpretować numerycznych wartości tych podobieństw.

2.3. Odległość na podstawie głosowania

Teraz przedstawimy sposób weryfikacji, czy wyborcy rzeczywiście chętniej głosują na podobne projekty. W tym celu, dla każdej pary projektów $c_i, c_j \in C$ proponujemy inną miarę dystansu[3] pomiędzy nimi jako odległość Jaccarda ich zbiorów poparcia:

$$d_W(c_i, c_j) := J(S(c_i), S(c_j))$$

Intuicyjnie, jeśli istotna część wyborców popierających zarówno projekt c_i, jak też c_j, pokrywa się, to projekty te określimy jako bliskie sobie. W ten sposób możliwe jest stwierdzenie, czy głosowanie na pewien projekt istotnie koreluje z głosowaniem na inne.

2.4. Badanie korelacji

W pierwszej kolejności należało ustalić, czy istnieje jakakolwiek statystycznie istotna korelacja pomiędzy dwoma opisanymi powyżej sposobami określania dystansu między projektami. W celu znalezienia odpowiedzi na to pytanie, zbadaliśmy dane z plików dostępnych na stronie `pabulib.org`. Mając ustaloną instancję wyborczą, dla każdego z projektów wygenerowaliśmy dwa wektory jego odległości od pozostałych projektów. Jeden z nich zawierał odległości lingwistyczne, drugi zaś te obliczone na podstawie głosowania. Formalnie, dla każdego $c_i \in C$ wyznaczyliśmy dwa wektory:

$$W(c_i) := [d_W(c_i, c_j)]_{j \neq i}, L(c_i) := [d_L(c_i, c_j)]_{j \neq i}$$

Następnie wyznaczyliśmy współczynniki korelacji Pearsona pomiędzy tymi wektorami. Należało uprzednio sprawdzić, czy spełnione są założenia, pod którymi wartości tych współczynników można interpretować.

[3]Tym razem indeks dolny W pochodzi od "wyborczy"

Po pierwsze, badane zmienne muszą być ciągłe bądź quasi-ciągłe – odległości pomiędzy projektami spełniają to założenie, gdyż nie są zmiennymi jakościowymi, lecz ilościowymi z interpretowalną średnią.

Po drugie, powinno być spełnione założenie o normalności rozkładu obu typów dystansów. W toku analizy okazało się, że część instancji wyborczych nie spełnia tego założenia. Napisaliśmy zatem program, który zidentyfikował wszystkie instancje, dla których założenie o normalności rozkładu jest spełnione na poziomie istotności $\alpha = 10\%$.

Opiszemy teraz krótko metodę wyszukiwania projektów, które spełniają założenie o normalności rozkładu.

2.4.1. Test statystyczny Jarque-Bera

Pierwsza faza weryfikacji polegała na wielokrotnym przeprowadzeniu testu statystycznego Jarque-Bera. Hipotezą zerową w tym teście jest

$$H_0: \text{ dane mają rozkład normalny } \mathcal{N}\left(\mu, \sigma^2\right) \text{ dla pewnych } \mu \in \mathbb{R}, \sigma^2 > 0$$

zaś hipotezą alternatywną jest

$$H_1: \text{ dane nie mają rozkładu normalnego}$$

Każdemu projektowi poddanemu pod głosowanie odpowiadają dwa wektory dystansów. Instancji wyborczych w bazie danych jest 730, każda z nich zawiera oczywiście wiele projektów. W celu weryfikacji, które z instancji wyborczych można poddać dalszej analizie, napisaliśmy kod oceniający rozkłady projektów zgłoszonych do danej instancji. Przykładowy efekt działania kodu dla pojedynczej instancji przedstawiają poniższe tabele:

	Poland, Warszawa, Ursynów 2019
Liczba projektów	58
p > 0.1	49
0.05 < p <= 0.1	3
0.01 < p <= 0.05	0
0.001 < p <= 0.01	2
p <= 0.001	4

(a) Tabela dla wektora dystansów Jaccarda

	Poland, Warszawa, Ursynów 2019
Liczba projektów	58
p > 0.1	36
0.05 < p <= 0.1	8
0.01 < p <= 0.05	4
0.001 < p <= 0.01	6
p <= 0.001	4

(b) Tabela dla wektora dystansów lingwistycznych

Rysunek 2.2: Przykład działania kodu do oceny normalności rozkładu - wybory na Ursynowie w roku 2019.

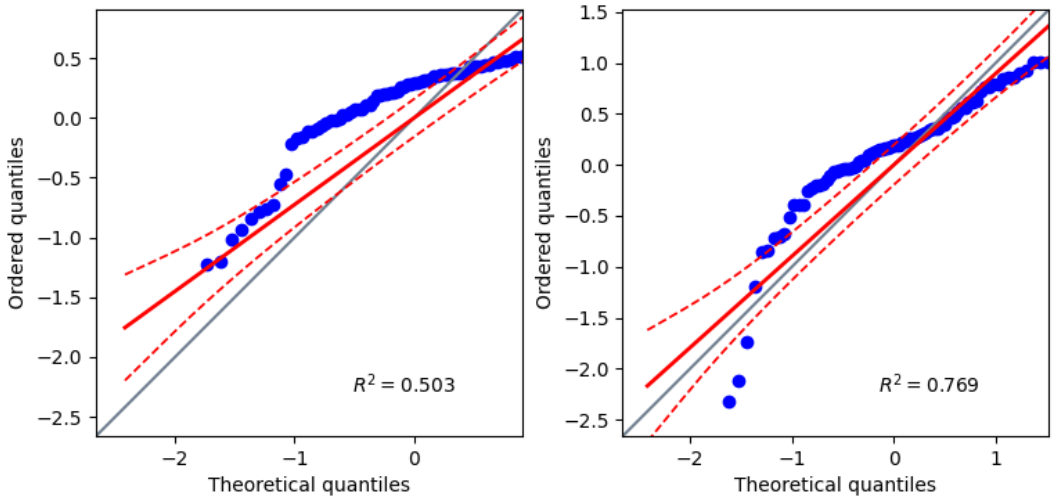
W powyższej tabeli oznaczenie p oznacza p-value pochodzące z testu Jarque-Bera. Tytułem krótkiego przypomnienia, p-value w teście statystycznym oznacza najmniejszy poziom ufności taki, przy którym nie ma podstaw do odrzucenia hipotezy zerowej. Oznacza to, że gdy p-value jest niższe niż przyjęty poziom ufności, należy odrzucić hipotezę zerową kosztem hipotezy alternatywnej. Stąd, w przypadku warszawskiego Ursynowa w roku 2019, przy testowaniu wektorów dystansów Jaccarda dla 49 spośród 58 projektów – co stanowi $84,48\%$ wszystkich projektów – otrzymaliśmy p-value powyżej 10%, zatem brak był podstaw do odrzucenia hipotezy o normalności rozkładu. W pozostałych przypadkach p-value było mniejsze niż 10%, zatem były podstawy do odrzucenia tej hipotezy. Dla dystansów Jaccarda 36 na 58 projektów spełniło to założenie, co stanowi około 62.07% wszystkich projektów. Analogicznie, w pozostałych przypadkach były podstawy do odrzucenia H_0.

Identyczne tabele zostały wygenerowane dla każdego z plików dostępnych w bazie, następnie zaś wyniki zostały zagregowane. Zgodnie z przyjętym poziomem istotności, jako instancje spełniające założenia uznane zostały takie, dla których stosunek liczby odrzuceń H_0 (w tabelach odpowiadało to p-value poniżej 10%) do liczby wszystkich projektów nie przekroczył 10%. Takich instancji było 170. Następne wyzwanie polegało na tym, że liczba projektów wśród wielu spośród tych instancji była niewielka – 136 spośród nich posiadało zgłoszonych mniej niż 15 projektów. Zdecydowaliśmy o ich odrzuceniu, gdyż przy tak małej próbie ciężko odróżnić szum bądź losowość od rzeczywistej, choćby przybliżonej normalności rozkładu. Ostatecznie wyłoniona została grupa 34 instancji, dla których przeprowadziliśmy dalszą analizę. Były to instancje spełniające założenia niezbędne do interpretowania współczynników korelacji Pearsona, ponadto zaś dostatecznie duża liczba zgłoszonych w nich projektów umożliwiała wyciągnięcie dalszych interesujących wniosków. Godną odnotowania, choć pozbawioną znaczenia w kontekście dalszej analizy ciekawostką jest fakt, że wszystkie wybory spełniające założenia odbyły się w Warszawie.

2.4.2. Wykresy kwantyl-kwantyl ($Q-Q$ plots)

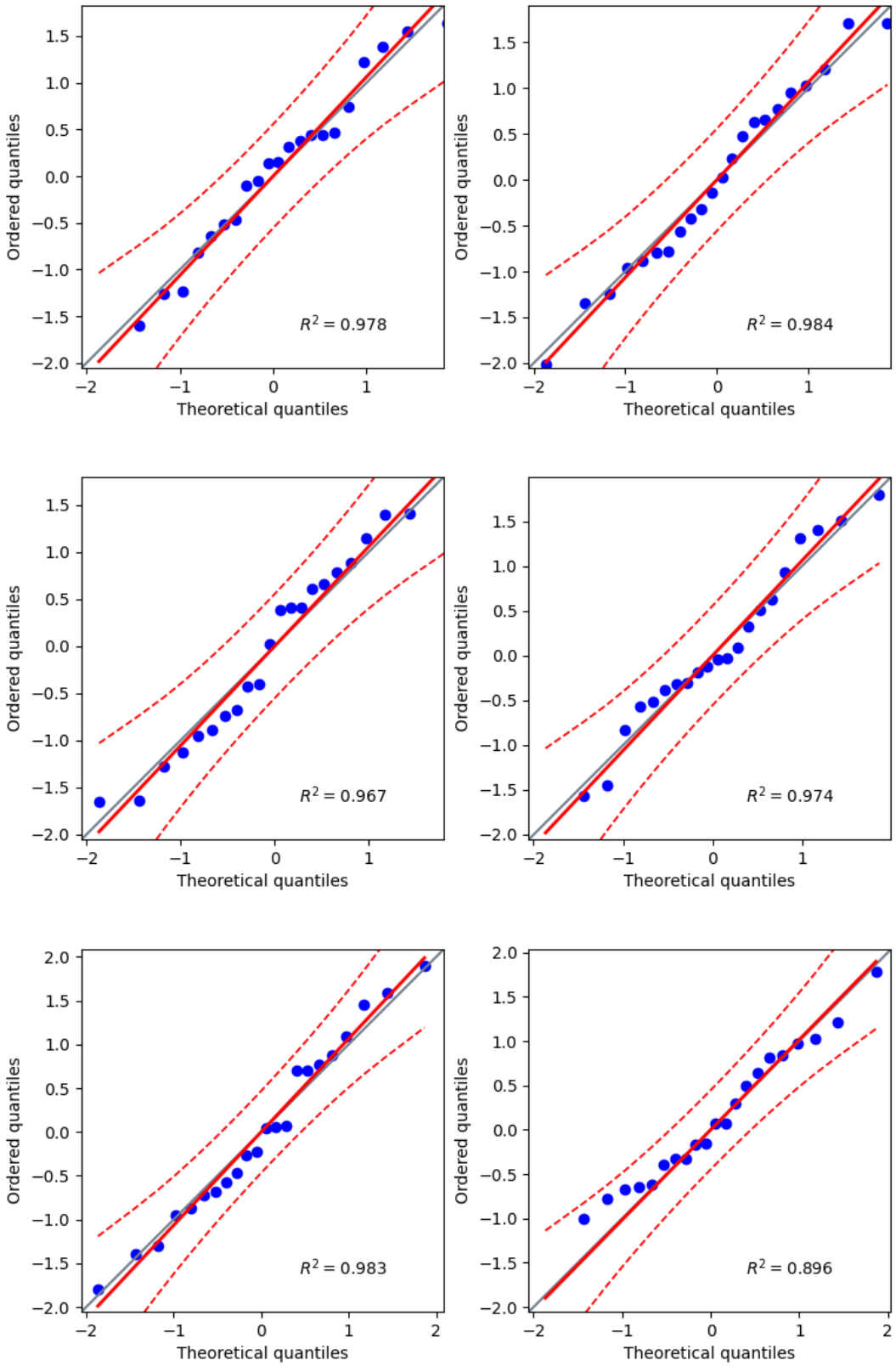
Aby potwierdzić poprawność wniosków płynących z wykonywania testów Jarque-Bera, w toku dalszej analizy posłużyliśmy się wykresami kwantyl-kwantyl. Wykresy te zestawiają ze sobą empiryczne kwantyle pochodzące z danych z teoretycznymi kwantylami rozkładu normalnego. Wbudowana funkcja najpierw sortuje wartości od najmniejszej do największej. Następnie obliczane są kwantyle odpowiadające każdej z wartości, po czym wartościom przypisane zostają kwantyle tego samego rzędu, który dana wartość wyznacza w danych, lecz pochodzące z rozkładu normalnego. Wówczas taki punkt jest nanoszony na wykres.

Przyjmuje się, że gdy niebieskie punkty układają się wzdłuż zaznaczonej prostej oraz leżą wewnątrz naniesionego na wykres przedziału ufności, dane mają rozkład normalny. Wykresy dla wszystkich projektów z wybranej grupy 34 instancji wyborczych znajdują się na naszym Githubie[4], poniżej zaś zamieszczamy jedynie kilka z nich, w celu poglądowego zilustrowania zastosowanej metody badawczej.

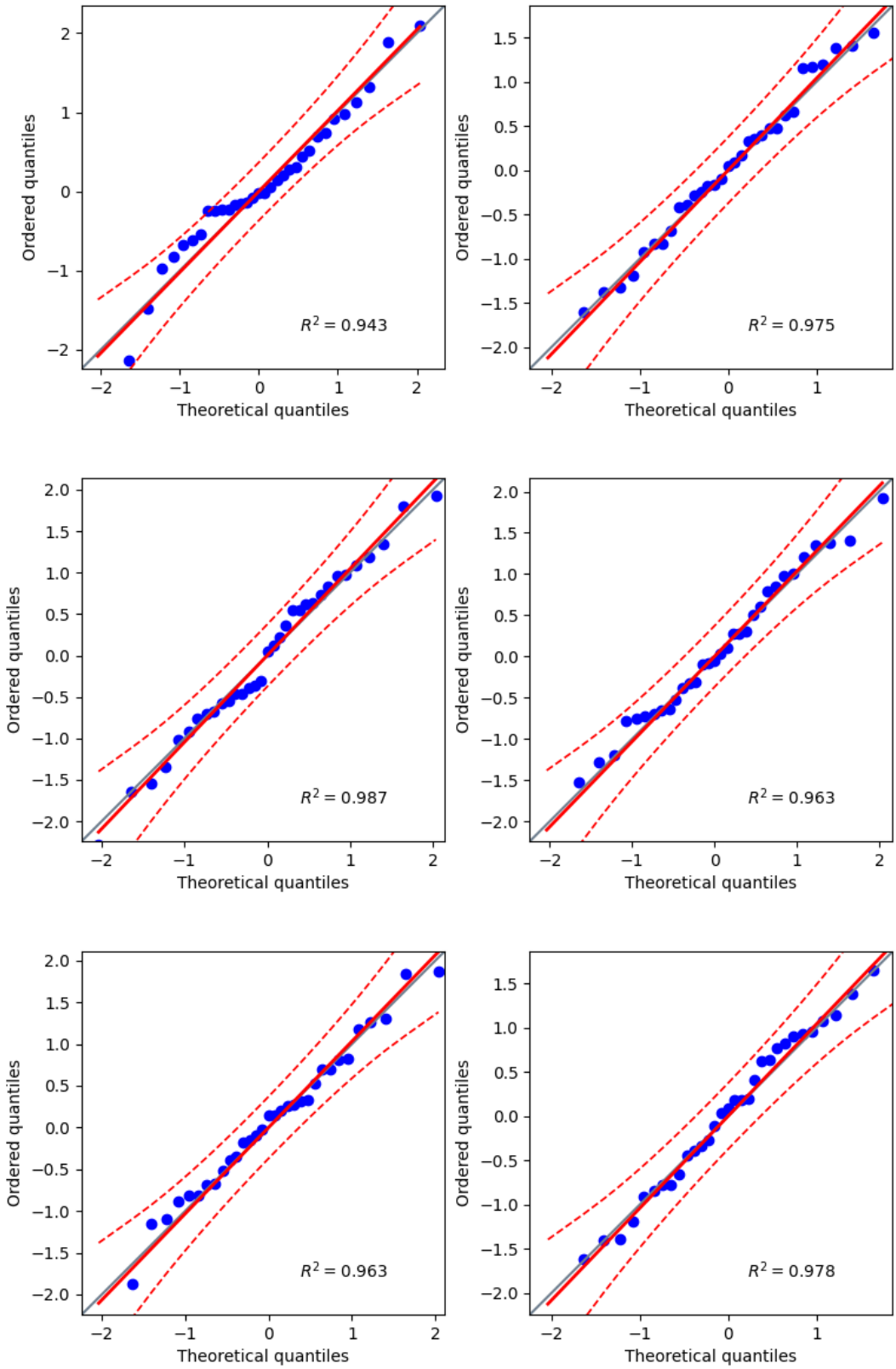


Rysunek 2.3: Przykładowe wykresy kwantyl-kwantyl wskazujące na brak normalności rozkładu. Powyższe wykresy wykonane zostały dla jednego z projektów poddanych pod głosowanie w Częstochowie w roku 2020. Wykres po lewej stronie dotyczy rozkładu wektora $W(c_i)$, zaś po prawej $L(c_i)$.

[4] `https://github.com/jakubweglowski/computational-social-choice`



Rysunek 2.4: Przykładowe wykresy kwantyl-kwantyl wykonane dla trzech projektów poddanych pod głosowanie na Żoliborzu w roku 2017. Wykresy po lewej stronie dotyczą rozkładów wektorów $W(c_i)$, zaś po prawej $L(c_i)$. Założenie o normalności rozkładu jest tutaj spełnione.



Rysunek 2.5: Przykładowe wykresy kwantyl-kwantyl wykonane dla trzech projektów poddanych pod głosowanie na Gocławiu w roku 2018. Wykresy po lewej stronie dotyczą rozkładów wektorów $W(c_i)$, zaś po prawej $L(c_i)$. Założenie o normalności rozkładu jest tutaj spełnione.

Ostatecznie we wszystkich instancjach spełniających założenia znalazły się łącznie 673 projekty. Średnia wartość współczynnika korelacji Pearsona wyniosła w przybliżeniu 0.738 przy odchyleniu standardowym wynoszącym około 0.110. Dokładnie 646 współczynników korelacji okazało się istotnych na poziomie $\alpha = 10\%$, co daje około 97% wszystkich.

Na podstawie dokonanej analizy uprawniony wydaje się wniosek, że istnieje statystycznie istotna, dodatnia korelacja pomiędzy obiema metodami określania dystansu między projektami. Daje to podstawy do dalszej analizy, ponieważ skoro korelacja istnieje, to zasadną jest próba poszukiwania głębszych związków pomiędzy tymi metrykami. Przykładowo, mając wyróżnioną grupę projektów o większym niż średnia dla instancji wyborczej podobieństwie Jaccarda, można spodziewać się także większego od średniej podobieństwa lingwistycznego w ramach tej grupy.

2.5. Klastrowanie

Do pracy nad poniższą sekcją używaliśmy biblioteki *scikit-learn* w dostępnej w języku *Python* [4].

2.5.1. Metoda K-średnich

W celu znalezienia rozbicia zbioru projektów na klastry, w pierwszej kolejności zastosowaliśmy algorytm K-średnich. Jest to algorytm szeroko stosowany w uczeniu maszynowym, którego działanie polega na znajdowaniu rozbicia zbioru danych na zadaną z góry liczbę K podzbiorów (klastrów). Celem jest znalezienie rozbicia, które minimalizuje wariancję w obrębie każdego z podzbiorów tworzących podział.

Formalnie, niech $(x_1, ..., x_n)$ oznacza wektor obserwacji, z których każda także może być wektorem: $x_i \in \mathbb{R}^d$. Algorytm rozpoczyna działanie od wylosowania K punktów zwanych *centroidami*: $(\mu_1, ..., \mu_K), \mu_i \in \mathbb{R}^d$, czyli punktów, które docelowo będą punktami wyznaczającymi średnie dla każdego z klastrów. Następnie powtarzane są następujące dwa kroki:

1. Zbiór danych dzielony jest na klastry w taki sposób, że punkt x_i staje się elementem klastra o środku μ_j, jeśli zachodzi warunek
$$\|x_i - \mu_j\| = \min_k \|x_i - \mu_k\|$$

2. Dla każdego z klastrów $S_j, j \in \{1, ..., k\}$, obliczane są nowe centroidy:
$$\mu_j = \frac{1}{|S_j|} \sum_{x \in S_j} x$$

Algorytm kończy działanie, gdy pod wpływem wykonywania kolejnych kroków przestają zachodzić zmiany w podziale całego zbioru na podzbiory. Ostatecznie, celem działania algorytmu K-średnich jest wybranie rozbicia $S = (S_1, ..., S_K)$ zbioru obserwacji $\{x_1, ..., x_n\}$, które minimalizuje sumę kwadratów długości różnic wektorów obserwacji oraz centroidu. Szukamy zatem:

$$\text{argmin}_S \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$

W tym miejscu należy zaznaczyć, że algorytm ten ma zasadniczą wadę – jest nią brak gwarancji osiągnięcia minimum globalnego opisanego powyżej zagadnienia minimalizacji wariancji. Algorytm ten zbiega jedynie do pewnego minimum lokalnego. Początkowe centroidy są generowane losowo właśnie po to, by zniwelować szansę nieosiągnięcia minimum globalnego. Dlatego w pracy nie poprzestaliśmy na przeprowadzeniu

pojedynczego klastrowania tym algorytmem, lecz dla każdej instancji przeprowadziliśmy 15 niezależnych klastrowań, dla każdego z nich zapisując wyniki oraz następnie je agregując. W ten sposób zwiększyliśmy szansę osiągnięcia minimum globalnego zdefiniowanej funkcji celu w większej liczbie klastrowań.

2.5.2. Metoda DBscan

W drugiej kolejności wykorzystany został algorytm DBscan [6]. Jego zaletą jest brak konieczności narzucenia a priori liczby klastrów, na które chcemy podzielić zbiór projektów. Algorytm ten potrzebuje pobrać jednakże dwa inne zewnętrzne parametry: ε oraz min_obs. Żeby zrozumieć działanie tego algorytmu, potrzebujemy wprowadzić kilka definicji.

Definicja 4 *Punkt p ze zbioru danych nazwiemy rdzeniowym* (ang. core point)*, jeśli co najmniej min_obs punktów leży w odległości co najwyżej ε od niego, licząc wraz z punktem p.*

Definicja 5 *Punkt q nazwiemy bezpośrednio osiągalnym* (ang. directly reachable)*, jeśli jego odległość od pewnego punktu rdzeniowego p jest nie większa niż ε.*

Definicja 6 *Punkt q nazwiemy osiągalnym* (ang. reachable) *z punktu p, jeżeli istnieją punkty*

$$p = p_1, p_2, ..., p_{n-1}, p_n = q$$

takie, że p_{i+1} jest bezpośrednio osiągalny z punktu p_i.

Klaster w tej metodzie tworzy punkt rdzeniowy p wraz ze zbiorem wszystkich punktów osiągalnych z niego. Każdy klaster zawiera zatem co najmniej jeden punkt rdzeniowy oraz może zawierać wiele punktów, które nie są rdzeniowe.

Widzimy zatem, że metoda klastrowania DBscan opiera się intuicyjnie na gęstości zbioru danych. Klastry będą tworzone przez punkty, które są ułożone gęsto obok siebie, przy czym miara tej gęstości upakowania punktów w przestrzeni jest wyrażona przez wartości parametrów ε oraz min_obs.

W toku analizy przyjęliśmy $min\_obs = 2$. Okazało się, że dobranie odpowiedniej wartości parametru ε stanowi równie poważne wyzwanie, co określenie liczby klastrów dla algorytmu K-means. Algorytm DBscan działa bardzo dobrze, jeśli dane wykazują wyraźną strukturę przestrzenną, natomiast jego działanie nie jest zadowalające gdy dane są zaszumione. Analiza działania DBscan na naszych danych wskazuje na to, że dane dotyczące dystansu lingwistycznego oraz wyborczego dystansu Jaccarda nie posiadają wyraźnej struktury przestrzennej, zaś swoją strukturą przypominają raczej losowy szum.

Metoda prób i błędów wskazała, że optymalne wartości parametru ε wahają się od 0.75 do 0.95 zależnie od instancji wyborczej. Dla bardzo wielu wartości parametru ε zwracane były jednakże jedynie dwa klastry, przy czym w jednym z nich znajdowały się trzy projekty, zaś cała reszta traktowana była jako szum i zwracana była jako osobny klaster. Pojawiła się zatem potrzeba zautomatyzowania procesu poszukiwania optymalnej wartości ε, tak, aby być w stanie przeszukać więcej możliwości niż przy pomocy metody prób i błędów.

Aby rozwiązać ten problem stworzyliśmy kod, którego zasada działania jest następująca: rozpoczynamy z parametrami $min = 0.5$ oraz $max = 1.5$. Dla i od 1 do 3:

1. Wykonujemy klastrowania dla wszystkich wartości ε od min do max z krokiem równym 10^{-i}.
2. Sprawdzamy, jaka była największa liczba klastrów, na jakie podzielony został zbiór projektów. Następnie wybieramy najmniejszą i największą wartość ε, dla której ta liczba klastrów została osiągnięta.

3. Nadpisujemy tę najmniejszą wartość jako min oraz największą wartość jako max, zwiększamy i o 1 i wracamy do punktu 1. listy.

Jako optymalną wartość ε wybieramy dowolną spośród tych, dla których po ostatniej iteracji liczba klastrów była największa. W ten sposób praktycznie gwarantujemy sobie, że nie jesteśmy w stanie znaleźć klastrowania z podziałem na więcej klastrów niż tego, które zostało wybrane tą metodą. Intuicyjnie można ten algorytm rozumieć jako próbę odtworzenia możliwie jak najwyraźniejszej struktury przestrzennej danych, na których pracujemy.

2.5.3. Metoda Spectral

Następną z wykorzystanych przez nas metod będzie Spectral clustering [9] [5]. Metoda ta jest silnie powiązana z innymi metodami klastrowania, gdyż jej główną ideą jest redukcja wymiaru danych, które następnie są klastrowane przy użyciu tychże metod.

Aby zrozumieć sposób działania metody Spectral, w pierwszej kolejności potrzebujemy zdefiniować macierz podobieństwa $A = (a_{ij})_{i,j=1,...,n}$, gdzie współczynniki a_{ij} są pewną miarą podobieństwa obserwacji x_i oraz x_j ze zbioru danych. Narzuca się warunek, żeby miara tego podobieństwa była symetryczna, tak aby macierz A była symetryczna. Następnie definiuje się tzw. laplasjan jako:

$$L := D - A$$

Gdzie $D = (d_{ij})_{i,j=1,...,n}$ jest macierzą diagonalną taką, że $d_{ii} = \sum_{j=1}^{n} a_{ij}$. Wartość d_{ii} jest więc sumą podobieństw obserwacji x_i do wszystkich innych obserwacji w zbiorze danych, włącznie z podobieństwem x_i do samej siebie. Najbardziej podstawowa wersja algorytmu Spectral polega na klastrowaniu zbioru danych, którego elementami są wektory własne laplasjanu odpowiadające kilku jego najmniejszym dodatnim wartościom własnym. Warto w tym miejscu zauważyć, że taki schemat postępowania jest dobrze zdefiniowany, gdyż macierz L jest symetryczna. Macierze symetryczne są diagonalizowalne, a ponadto jeśli ich elementami są wyłącznie liczby rzeczywiste, mają one zawsze rzeczywiste wartości własne. Samo klastrowanie zbioru danych złożonego z wektorów własnych jest wykonywane przy użyciu innych popularnych metod klastrowania, takich jak K-means czy DBscan.

W praktyce jednak nie wykorzystuje się wektorów własnych samego laplasjanu, lecz uprzednio dokonuje się jego normalizacji. Celem normalizacji jest takie wyskalowanie laplasjanu, aby $l_{ii} = 1$ dla wszystkich $i \in \{1, ..., n\}$. Jedną z metod normalizacji, o których warto w tym miejscu wspomnieć, to algorytm Shi-Malika [7], wykorzystujący wektory i wartości własne symetrycznego znormalizowanego laplasjanu:

$$L^{norm} := Id_n - D^{-1/2} A D^{-1/2}$$

gdzie pisząc $D^{-1/2}$ mamy na myśli taką macierz odwracalną X, że $\left(X^{-1}\right)^2 = D$. Innym powszechnie stosowanym przekształceniem jest random walk, które definiuje się jako:

$$L^{rw} := D^{-1} L = Id_n - D^{-1} A$$

2.5.4. Losowy podział zbioru projektów

Aby stwierdzić, czy klastrowanie opisanymi powyżej metodami rzeczywiście daje istotne rezultaty, przeprowadziliśmy także klastrowanie losowe. Schemat postępowania był silnie związany z wynikami zwracanymi przez algorytm K-średnich. Algorytm ten zwracał rozbicie zbioru projektów na K podzbiorów. Każdy z tych podzbiorów zawierał oczywiście pewną liczbę projektów.

Klastrowanie losowe oparte zostało na następującym pomyśle – na początku lista projektów została losowo pomieszana. Później zaś losową listę projektów dzieliliśmy na klastry takiej liczebności, jakiej były

kolejne klastry zwrócone przez algorytm K-means. Przykładowo, gdy algorytm ten zwrócił 3 klastry o liczebnościach kolejno $2, 5$ oraz 3, to podział losowy polegałby na umieszczeniu pierwszych dwu projektów losowej listy w pierwszym podzbiorze, kolejnych pięciu w drugim, zaś ostatnich trzech w trzecim. Na każde klastrowanie K-means wykonanych zostało 100 klastrowań losowych. Dzięki temu nie tylko uwzględniona została losowość klastrowania, jak też porównywalność z wynikami zwracanymi przez K-means.

2.5.5. Wyniki klastrowania

Przedstawimy teraz wyniki otrzymanej analizy. Podobnie jak w przypadku analizy korelacji, ogólne wyniki zostały uzyskane poprzez agregację analiz dla poszczególnych instancji wyborczych, stąd rozpoczniemy od omówienia metody użytej do badania pojedynczych wyborów. Ogólny schemat postępowania był następujący:

1. Wykonujemy klastrowanie w oparciu o dystans lingwistyczny bądź wyborczy Jaccarda, program zwraca podział zbioru projektów na klastry. Oznaczając przez K liczbę klastrów, otrzymujemy rozbicie $\mathbb{S} = \{S_1, ..., S_K\}$ zbioru projektów C na klastry.

2. Obliczamy przy pomocy algorytmu lingwistycznego średnie podobieństwo nazw projektów w obrębie każdego klastra, tj. dla każdego $k \in \{1, ..., K\}$ oznaczamy przez $n_k := |S_k|$ i obliczamy:

$$\bar{L}_k := \frac{1}{\frac{n_k(n_k-1)}{2}} \sum_{c_i, c_j \in S_k : i<j} (1 - d_L(c_i, c_j))$$

3. Jako ocenę jakości pojedynczego klastrowania $\mathbb{S} = \{S_1, ..., S_K\}$ przyjmiemy liczbę:

$$Q(\mathbb{S}) := \frac{\sum_{k=1}^{K} \bar{L}_k \cdot n_k}{\sum_{k=1}^{K} n_k}$$

   Jest to średnia ważona podobieństw w obrębie klastrów (obliczonych w punkcie 2.), w której wagami są liczby projektów w klastrach.

4. W przypadku, gdy wykonujemy wiele klastrowań, jako ocenę jakości całego procesu przyjmujemy średnią arytmetyczną oceny pojedynczych klastrowań. Jeśli mamy dane klastrowania $\{\mathbb{S}_1, ..., \mathbb{S}_N\}$, ocena ta będzie wyrażać się wzorem:

$$Q\left(\{\mathbb{S}_1, ..., \mathbb{S}_N\}\right) := \frac{1}{N} \sum_{k=1}^{N} Q\left(\mathbb{S}_k\right)$$

Aby lepiej zrozumieć powyższy schemat, omówimy go na konkretnym przykładzie wyborów, które odbyły się na warszawskim Zaciszu w roku 2017. Pod głosowanie poddanych zostało wówczas 21 projektów. Spójrzmy najpierw na wyniki klastrowania K-means z liczbą klastrów $K = 5$ na obu typach dystansów:

```
Klaster nr 1:
Studzienka odwadniająca na ulicy Rozwadowskiej.
Progi zwalniające na Targówku, ul. Krośniewicka
Remont chodnika na ulicy Rozwadowskiej.

Klaster nr 2:
Gimnastyka na świeżym powietrzu przy Domu Kultury Zacisze
Strefa wolnego handlu
Podwieczorki z kulturą dla dzieci (Zacisze)
Angielski z native speakerem dla mieszkańców Zacisza
Wyrównywanie szans w rozwoju psychomotorycznym u dzieci
Sąsiedzkie warsztaty kulinarne, czyli pasja tworzenia smaków na Zaciszu
Wspieraj swoje dziecko! Ocenianie kształtujące dla rodziców! - Targówek

Klaster nr 3:
Doświetlenie przejść dla pieszych prowadzących do lasu Bródnowskiego na ul. Kondratowicza.
Doświetlenie przejścia dla pieszych na ul. Codziennej róg ul. Blokowej
Oświetlenie skweru przy ul. Św. Wincentego i Kondratowicza
Wymiana opraw oświetleniowych na skrzyżowaniach: Gilarska-Samarytanki oraz
        Jórskiego-Samarytanki wraz z modernizacją szafy zasilającej.
Zielony mural przy Lesie Bródnowskim
Organizacja I-ego Biegu Zacisza 14 maja 2017r.

Klaster nr 4:
Szklana góra - wielofunkcyjne przeszklone pomieszczenie przy DK Zacisze.
Dzieciaki nie płac(z)ą - bezpłatne koncerty i warsztaty artystyczne dla dzieci na Zaciszu.
Muzycznie dla sąsiadów - cykl 10 bezpłatnych koncertów na Zaciszu
Zespół teatralno-kabaretowy dla seniorów

Klaster nr 5:
Oświetlenie boiska i monitoring przy ul. Blokowej 3
```

Rysunek 2.6: Przykładowe klastrowanie K-means po dystansach wyborczych Jaccarda dla wyborów na Zaciszu w 2017 roku. Ocena lingwistyczna tego klastrowania wyniosła w przybliżeniu 0.520.

```
Klaster nr 1:
Oświetlenie boiska i monitoring przy ul. Blokowej 3
Doświetlenie przejść dla pieszych prowadzących do lasu Bródnowskiego na ul. Kondratowicza.
Doświetlenie przejścia dla pieszych na ul. Codziennej róg ul. Blokowej

Klaster nr 2:
Dzieciaki nie płac(z)ą - bezpłatne koncerty i warsztaty artystyczne dla dzieci na Zaciszu.
Muzycznie dla sąsiadów - cykl 10 bezpłatnych koncertów na Zaciszu
Zespół teatralno-kabaretowy dla seniorów
Podwieczorki z kulturą dla dzieci (Zacisze)
Angielski z native speakerem dla mieszkańców Zacisza
Wyrównywanie szans w rozwoju psychomotorycznym u dzieci
Sąsiedzkie warsztaty kulinarne, czyli pasja tworzenia smaków na Zaciszu
Wspieraj swoje dziecko! Ocenianie kształtujące dla rodziców! - Targówek

Klaster nr 3:
Szklana góra - wielofunkcyjne przeszklone pomieszczenie przy DK Zacisze.
Gimnastyka na świeżym powietrzu przy Domu Kultury Zacisze
Oświetlenie skweru przy ul. Św. Wincentego i Kondratowicza
Zielony mural przy Lesie Bródnowskim
Studzienka odwadniająca na ulicy Rozwadowskiej.
Progi zwalniające na Targówku, ul. Krośniewicka
Remont chodnika na ulicy Rozwadowskiej.

Klaster nr 4:
Wymiana opraw oświetleniowych na skrzyżowaniach: Gilarska-Samarytanki oraz
        Jórskiego-Samarytanki wraz z modernizacją szafy zasilającej.

Klaster nr 5:
Strefa wolnego handlu
Organizacja I-ego Biegu Zacisza 14 maja 2017r.
```

Rysunek 2.7: Przykładowe klastrowanie K-means po dystansach lingwistycznych dla wyborów na Zaciszu w 2017 roku. Ocena lingwistyczna tego klastrowania wyniosła w przybliżeniu 0.571.

Teraz spójrzmy, jak przedstawiają się wyniki analizy dla różnej liczby klastrów K:

Liczba klastrów (K)	**Linguistic**	**K-means**	**Spectral**	**DBscan**	**Random**
1	0.46632	0.46632	0.46632	NA	0.46632
2	0.49983	0.49387	0.48708	NA	0.46665
3	0.51577	0.50299	0.49878	0.48856	0.46673
4	0.52687	0.50792	0.50394	NA	0.46663
5	0.53669	0.51116	0.50963	NA	0.46658
6	0.54471	0.51367	0.51279	NA	0.46692
7	0.55197	0.51714	0.51841	NA	0.46714
8	0.55883	0.52087	0.52212	NA	0.46748
9	0.56498	0.52479	0.52483	NA	0.46791
10	0.571	0.52808	0.52763	NA	0.46864
11	0.57599	0.53152	0.53037	NA	0.46935
12	0.58053	0.53478	0.53434	NA	0.47013
13	0.5841	0.5373	0.53765	NA	0.47109
14	0.58664	0.53887	0.54027	NA	0.47199
15	0.58841	0.5405	0.5416	NA	0.47284

Rysunek 2.8: Podsumowanie oceny jakości klastrowania dla wyborów na warszawskim Zaciszu w roku 2017.

Liczba klastrów (K)	**Linguistic**	**K-means**	**Spectral**	**Random**
2	0.03351	0.02755	0.02076	0.00106
3	0.03541	0.02602	0.02369	0.00147
4	0.03646	0.0241	0.02238	0.00188
5	0.03749	0.02269	0.02303	0.00261
6	0.0391	0.02186	0.02248	0.00363
7	0.0404	0.02207	0.02444	0.00388
8	0.04195	0.02288	0.02492	0.00438
9	0.04306	0.02425	0.0249	0.00447
10	0.04468	0.02513	0.0251	0.0053
11	0.04585	0.0263	0.02569	0.00584
12	0.04683	0.02767	0.02787	0.00617
13	0.04685	0.02827	0.02912	0.00661
14	0.04606	0.02815	0.02959	0.00725
15	0.04506	0.02782	0.02901	0.00776

Rysunek 2.9: Odchylenia standardowe ocen jakości klastrowania dla wyborów na warszawskim Zaciszu w roku 2017.

W Tabeli 2.8 przedstawiono uśrednione oceny podobieństwa lingwistycznego projektów w zwracanych przez poszczególne algorytmy klastrach. K-means i Spectral potrzebują pobrać liczbę klastrów jako parametr, więc dla nich wykonane zostały klastrowania z liczbą klastrów od 1 do 15. Dla DBscan wykonane zostało tylko jedno klastrowanie, gdyż algorytm zawsze zwraca ten sam wynik. W kolumnie "DBscan" widać, ile klastrów zostało uzyskanych przez opisany w Sekcji 2.5.2 algorytm, gdyż dla pozostałych liczb klastrów nie ma zapisanego wyniku.

W kolumnie "Linguistic" Tabeli 2.8 znajduje się ocena klastrowania po odległościach lingwistycznych przy użyciu algorytmu K-means. Klastrowanie po odległościach lingwistycznych odbywało się z wykorzystaniem K-means, ponieważ algorytm Spectral zwracał praktycznie identyczne wyniki, zaś algorytm DBscan nie byłby dobrym wyborem ze względu na brak możliwości sterowania liczbą otrzymanych klastrów. Wyniki zapisane w tabeli to średnia z $N = 15$ wykonanych klastrowań, zgodnie z punktem 4. schematu ogólnego zawartego powyżej.

W następnych trzech kolumnach Tabeli 2.8 zapisane zostały oceny klastrowania po odległościach wyborczych Jaccarda. Tak jak w przypadku klastrowania po dystansach lingwistycznych, zarówno dla algorytmu K-means, jak i dla algorytmu Spectral, dla każdej liczby klastrów wykonanych zostało $N = 15$ klastrowań. Oceny zapisane w tabeli to średnia ocen tych klastrowań, zgodnie z punktem 4. ogólnego schematu opisanego powyżej.

W ostatniej kolumnie zapisane zostały oceny klastrowania losowego, czyli średnie podobieństwa w obrębie klastrów, jeśli klastrowanie jest losowym podziałem zbioru projektów. Schemat klastrowania losowego opisany jest bliżej w Sekcji 2.5.4. W tym miejscu przypomnimy jedynie, że na każde klastrowanie K-means wykonywanych było 100 klastrowań losowych, co łącznie daje $N = 1500$ losowych klastrowań na jedną instancję wyborczą. Podobnie jak w przypadku pozostałych algorytmów, wyniki zapisane w tabeli są średnimi z pojedynczych ocen.

Jak wspomnieliśmy powyżej, do oceny jakości klastrowania użyliśmy algorytmu lingwistycznego. Podstawowym celem klastrowania jest znalezienie podzbiorów zbioru obserwacji, które wykazują się wyższym podobieństwem niż przeciętne podobieństwo obserwowane w danych. Jeśli więc klastrowanie odbywa się na danych dotyczących dystansów lingwistycznych, projekty w obrębie klastrów cechują się większym niż przeciętne podobieństwem swoim nazw (opisów). Stosowanie algorytmu lingwistycznego do oceny niesie więc za sobą ważną konsekwencję – klastrowanie na danych dotyczących dystansów lingwistycznych jest oceniane tym samym algorytmem, który został użyty do stworzenia tych danych. Oceny jakości klastrowań po dystansach lingwistycznych będą zatem wyższe, niż oceny pochodzące z klastrowania po dystansach wyborczych Jaccarda. Z tego powodu liczby w kolumnie "Linguistic" wyznaczają swego rodzaju punkt odniesienia – górną granicę oceny jakości klastrowania po dystansach wyborczych Jaccarda.

Tabela 2.9 zestawia odchylenia standardowe ocen klastrowań odpowiadające średnim zamieszczonym w Tabeli 2.8. Miara ta nie niesie za sobą wartości informacyjnej dla liczby klastrów równej 1, gdyż wtedy każde klastrowanie jest identyczne i odchylenie standardowe wynosi 0. Podobnie, dla algorytmu DBscan nie podajemy odchyleń standardowych oceny jakości, gdyż przy jego użyciu wykonane zostało tylko jedno klastrowanie. Warto zauważyć, że klastrowania losowe charakteryzują się o rząd wielkości niższym odchyleniem standardowym niż pozostałe algorytmy. Oznacza to, że różne klastrowania losowe są do siebie bardziej podobne pod względem oceny ich jakości niż różne klastrowania wykonane dla innych algorytmów. Może to sugerować, że w przypadku K-means i Spectral trafiają się klastrowania z istotnie wyższą od średniej oceną jakości. Może to sugerować osiąganie przez te algorytmy opisanych wcześniej globalnych minimów zagadnień optymalizacyjnych. W drugą stronę, niektóre klastrowania osiągają jedynie minima lokalne, stąd mogą pochodzić oceny istotnie niższe od średniej.

Porównując ze sobą obie tabele, oceny klastrowania losowego są mniejsze od średnich ocen dla pozostałych algorytmów o więcej niż dwa odchylenia standardowe. Uprawnionym wydaje się być wniosek, że klastrowanie z wykorzystaniem K-means, Spectral i DBscan po dystansach wyborczych Jaccarda charakteryzuje się wyższym średnim podobieństwem nazw projektów niż klastrowanie losowe. Oznacza to, że podobieństwo projektów wynikające z samych wyników głosowania jest powiązane z podobieństwem ich nazw, zaś stopień tego powiązania jest wyraźnie większy niż przypadkowy. Wniosek, który wyciągamy z analizy dla wyborów na warszawskim Zaciszu jest taki, że wyborcy chętniej głosowali wówczas na projekty

do siebie podobne.

Dla każdej z 34 instancji wyborczych spełniających konieczne założenia przeprowadzona została analogiczna analiza. Następnie wyniki zostały zagregowane, w efekcie czego powstała Tabela 2.10, którą zamieszczamy poniżej.

Instancja wyborcza	**Linguistic**	**K-means**	**Spectral**	**DBscan**	**Random**
Chrzanów, Jelonki 2017	0.454209	0.40998	0.40991	0.40316	0.40178
Sawa 2017	0.529858	0.49409	0.50504	0.46159	0.46203
Ursynów Wysoki Północny 2017	0.502278	0.45908	0.45739	0.42993	0.43491
Targówek 2018	0.457706	0.40375	0.40554	0.39728	0.40174
Grochów Centrum 2017	0.525805	0.46597	0.46345	0.46860	0.45469
Zacisze 2017	0.550176	0.51799	0.51705	0.48856	0.46843
Chrzanów, Jelonki 2018	0.481195	0.42705	0.43590	0.40115	0.42074
Ursus Północny 2018	0.501659	0.45851	0.46698	0.44206	0.44529
Rembertów 2019	0.600805	0.55333	0.55594	0.54268	0.52182
Zacisze 2018	0.485201	0.44175	0.44602	0.45341	0.40051
Czyste, Mirów 2017	0.539535	0.48848	0.49812	0.48644	0.47427
Służewiec 2019	0.542220	0.49248	0.49058	0.49478	0.47247
Wierzbno Wyględów 2019	0.465385	0.40704	0.39761	0.39991	0.39411
Saska Kępa 2018	0.505760	0.44486	0.45676	0.47813	0.43149
Gocław 2017	0.588641	0.54243	0.54477	0.52999	0.50080
Żoliborz Centralny 2017	0.492625	0.43730	0.43730	0.43159	0.41346
Targówek Mieszkaniowy 2019	0.531834	0.46700	0.47391	0.46123	0.46229
Żoliborz Centralny 2019	0.494772	0.44037	0.44437	0.41222	0.42594
Grochów Południowy 2017	0.595110	0.56282	0.57111	0.59967	0.51287
Saska Kępa 2019	0.529415	0.47593	0.47755	0.46679	0.44703
Wysokie Okęcie 2018	0.524025	0.47287	0.47596	0.46091	0.45592
Chomiczówka 2018	0.458317	0.41232	0.40992	0.41483	0.40393
Śródmieście 2019	0.504425	0.46342	0.46731	0.46024	0.42025
Gocław 2018	0.525041	0.49473	0.49630	0.50684	0.44903
Anin 2017	0.540159	0.43514	0.44547	0.42689	0.38836
Bródno 2019	0.483233	0.44352	0.44420	0.43224	0.41724
Wilanów 2021	0.489929	0.43889	0.44138	0.42999	0.42594
Wawrzyszew 2019	0.583323	0.54024	0.54371	0.55827	0.50215
Stare Włochy 2018	0.468609	0.43473	0.43103	0.41461	0.40420
Targówek Mieszkaniowy 2018	0.439873	0.39041	0.38827	0.41024	0.36594
Nowa Praga z Pelcowizną 2017	0.425549	0.37188	0.37469	0.35532	0.37827
Sady Żoliborskie-Zatrasie 2017	0.546304	0.47692	0.48362	0.47269	0.47886
Bródno-Podgrodzie 2017	0.509357	0.45725	0.46725	0.44132	0.44525
Marymont-Potok-Żoliborz 2017	0.478308	0.42717	0.42643	0.41769	0.41724
Średnia ocena	0.510313	0.460285	0.463260	0.454449	0.438213
Odchylenie standardowe	0.042518	0.044681	0.046129	0.049692	0.037709

Rysunek 2.10: Średnie oceny jakości klastrowania z podziałem na poszczególne instancje wyborcze

Instancja wyborcza	**Linguistic**	**K-means**	**Spectral**	**Random**
Chrzanów, Jelonki 2017	0.038491	0.01415	0.01491	0.00897
Sawa 2017	0.037614	0.02128	0.02967	0.00601
Ursynów Wysoki Północny 2017	0.039436	0.01896	0.01699	0.00557
Targówek 2018	0.034719	0.01244	0.01279	0.00920
Grochów Centrum 2017	0.043900	0.01381	0.01219	0.00834
Zacisze 2017	0.041677	0.02544	0.02517	0.00428
Chrzanów, Jelonki 2018	0.042581	0.01907	0.02482	0.01431
Ursus Północny 2018	0.037092	0.01719	0.01757	0.00586
Rembertów 2019	0.041326	0.01869	0.01984	0.00356
Zacisze 2018	0.046955	0.02530	0.02482	0.00727
Czyste, Mirów 2017	0.037348	0.01284	0.01749	0.00349
Służewiec 2019	0.037721	0.01638	0.01519	0.00615
Wierzbno Wyględów 2019	0.041584	0.01507	0.01019	0.00791
Saska Kępa 2018	0.043794	0.01743	0.02309	0.00644
Gocław 2017	0.044216	0.02780	0.03063	0.00352
Żoliborz Centralny 2017	0.049339	0.02154	0.02057	0.00722
Targówek Mieszkaniowy 2019	0.039044	0.01010	0.01305	0.00410
Żoliborz Centralny 2019	0.044383	0.01904	0.01779	0.00851
Grochów Południowy 2017	0.042722	0.02420	0.02742	0.00399
Saska Kępa 2019	0.044889	0.02765	0.02470	0.00630
Wysokie Okęcie 2018	0.038069	0.01738	0.01709	0.00579
Chomiczówka 2018	0.037972	0.01316	0.01376	0.01448
Śródmieście 2019	0.045731	0.03242	0.03524	0.01189
Gocław 2018	0.043438	0.02502	0.02553	0.00237
Anin 2017	0.075519	0.04622	0.04882	0.01634
Bródno 2019	0.042646	0.02142	0.02107	0.00552
Wilanów 2021	0.036326	0.01266	0.01327	0.00464
Wawrzyszew 2019	0.043158	0.02148	0.02413	0.00364
Stare Włochy 2018	0.043349	0.02913	0.02543	0.00980
Targówek Mieszkaniowy 2018	0.047086	0.02339	0.02409	0.01404
Nowa Praga z Pelcowizną 2017	0.039693	0.02545	0.02455	0.02131
Sady Żoliborskie-Zatrasie 2017	0.043324	0.01069	0.01262	0.00549
Bródno-Podgrodzie 2017	0.036527	0.01619	0.02387	0.00805
Marymont-Potok-Żoliborz 2017	0.037820	0.01090	0.01058	0.00849

Rysunek 2.11: Średnie odchylenia standardowe ocen jakości klastrowania dla poszczególnych instancji wyborczych.

Mając w pamięci wygląd Tabeli 2.8 dla wyborów na Zaciszu, możemy łatwo zrozumieć znaczenie powyższych danych. Najpierw dla każdej instancji wyborczej stworzono tabelę taką, jak dla Zacisza. Następnie z każdej kolumny takiej tabeli obliczono średnią i zapisano ją w Tabeli 2.10 w wierszu odpowiadającym danej instancji wyborczej i kolumnie odpowiadającej uśrednionej kolumnie. Dane zawarte w Tabeli 2.10 są zatem uśrednieniem analizy pojedynczej instancji po wszystkich liczbach klastrów[5]. Analogiczny zabieg przeprowadzono także dla tabel z odchyleniami standardowymi, w efekcie czego powstała Tabela 2.11.

[5]Dla DBscan nie dokonano uśrednienia, gdyż dla tego algorytmu liczba klastrów nie zmienia się. W tabeli zapisano jedyny dla każdej instacji uzyskany rezultat klastrowania.

2.6. Wnioski

Z Tabel 2.10 oraz 2.11 nie daje się wysnuć jednoznacznych wniosków. Dla niektórych analizowanych instancji wyborczych różnica w ocenie jakości klastrowania między algorytmami K-means, Spectral oraz DBscan na dystansach wyborczych Jaccarda, a klastrowaniem losowym jest wyraźna. Jako przykłady takich instancji można podać chociażby wybory na Śródmieściu w roku 2019 czy też omówione już obszernie powyżej Zacisze w roku 2017, gdzie różnica między średnią ocen klastrowań dla powyższych algorytmów a klastrowaniem losowym jest większa niż suma odchyleń standardowych tych ocen. Takie instancje wyborcze pozwalają pozytywnie odpowiedzieć na postawione w pracy pytanie badawcze – wyborcy rzeczywiście chętniej głosowali tam na projekty do siebie podobne.

Jednakże obecne są w tym zestawieniu również takie instancje, dla których algorytmy klastrowania oceniane są podobnie lub gorzej niż klastrowania losowe – jako przykłady można wskazać Nową Pragę z Pelcowizną czy też Marymont-Potok-Żoliborz w roku 2017. Dla takich instancji wyborczych odpowiedź na postawione pytanie wydaje się być negatywną – wyborcy prawdopodobnie nie głosowali tam na podobne projekty.

Uśrednienie uzyskanych rezultatów po wszystkich instancjach wyborczych wskazuje na to, że przeciętna ocena klastrowania z użyciem K-means, Spectral oraz DBscan jest wyższa od przeciętnej oceny klastrowania losowego. Średnia ocena klastrowania lingwistycznego wyniosła około 0.510, zaś oceny K-means, Spectral i DBscan na dystansach wyborczych Jaccarda wyniosły odpowiednio około $0.460, 0.463, 0.454$. Przeciętna ocena klastrowania losowego wyniosła w przybliżeniu 0.438. Odpowiadające tym wartościom odchylenia standardowe sugerują jednakże, że różnice te nie są istotne statystycznie, gdyż przeciętna ocena klastrowania losowego znajduje się w odległości jednego odchylenia standardowego od przeciętnych ocen dla pozostałych algorytmów.

Jednoznaczna odpowiedź na postawione pytanie: "Czy wyborcy głosują na podobne projekty?" jest bardzo trudna. Z powyższej analizy wynika, że dla wielu instancji wyborczych powinna być ona pozytywna, gdyż różnice w ocenach klastrowań są istotne biorąc pod uwagę średnie i odpowiadające im odchylenia standardowe. Dla niektórych instancji odpowiedź powinna być negatywna, gdyż różnice te nie są znaczące. Ostateczny wniosek, jaki naszym zdaniem płynie z wykonanej analizy, jest taki, że każdą z instancji wyborczych należy analizować indywidualnie pod kątem odpowiedzi na postawione pytanie.

Rozdział 3

Podsumowanie

Podsumowując, w niniejszej pracy spróbowaliśmy odpowiedzieć na pytanie, czy wyborcy w głosowaniach nad budżetem obywatelskim częściej głosują na podobne do siebie projekty. W tym celu zdefiniowaliśmy na zbiorze zgłoszonych projektów dwa sposoby mierzenia odległości między dwoma projektami – odległość lingwistyczną ich nazw (opisów) oraz odległość wyborczą Jaccarda, którą obliczaliśmy na podstawie końcowych wyników głosowania wyborców.

W dalszej kolejności sprawdziliśmy, czy istnieją statystyczne podstawy do poszukiwania zależności między tymi miarami dystansu. Wykorzystaliśmy do tego analizę korelacji liniowej z użyciem współczynników Pearsona, sprawdzając uprzednio założenia niezbędne do ich interpretacji. Należało sprawdzić między innymi, czy wektory wygenerowane dla obu typów dystansów kodujące odległość danego projektu od pozostałych projektów pochodzą z rozkładu normalnego. Aby sprawdzić to założenie, wykonaliśmy dla każdego projektu w bazie danych *Pabulib* testy statystyczne Jarque-Bera. Po agregacji wyników tej analizy użyliśmy wykresów kwantyl-kwantyl w celu potwierdzenia wniosków z niej płynących.

Przeprowadzona analiza wykazała, że istnieje statystycznie istotna, dodatnia korelacja pomiędzy miarą lingwistyczną a miarą wyborczą Jaccarda. W związku z tym w dalszej części pracy przeprowadziliśmy klastrowania zbiorów projektów w celu znalezienia ich podziału na klastry projektów podobnych do siebie – zarówno w sensie odległości lingwistycznej, jak też wyborczej Jaccarda. Później użyliśmy algorytmu lingwistycznego aby ocenić jakość wykonanego klastrowania. Jako górny skraj analizy jakości klastrowania posłużyła nam ocena klastrowania po dystansach lingwistycznych, zaś jako dolny skraj tej oceny przyjęliśmy klastrowanie losowe. Następnie przeanalizowaliśmy, gdzie na tym spektrum należy umieścić oceny lingwistyczne klastrowań wykonanych względem dystansów wyborczych Jaccarda przy użyciu powszechnie stosowanych w uczeniu maszynowym algorytmów K-means, Spectral oraz DBscan.

Analiza średnich ocen klastrowań wraz z odpowiadającym im odchyleniom standardowym wykazała, że dla wielu instancji wyborczych głosujący rzeczywiście chętniej akceptowali projekty podobne do siebie. Znalazły się jednak również i takie instancje, dla których różnice w ocenach były nieistotne statystycznie. Ponieważ końcowa agregacja wyników dla pojedynczych instancji nie dała jasnej odpowiedzi na postawione pytanie badawcze, dalsze analizy tego typu należy według nas przeprowadzać indywidualnie dla każdej instancji wyborczej.

W Sekcji 2.5.2 poświęconej omówieniu sposobu działania algorytmu DBscan wspomnieliśmy, że obserwacje wyników klastrowania zwracanych przez ten algorytm wskazują na silne zaszumienie danych dotyczących obu miar dystansu. Naturalnym krokiem kontynuacji podjętej ścieżki badań jest znalezienie sposobu na identyfikację grup projektów, które w rzeczywistości stanowią szum. Wówczas można byłoby kontynuować

badania jedynie dla projektów, które nie są szumem – dla grup, dla których być może istnieją rzeczywiste, silniejsze zależności, korelacje i związki niż te, które udało się nam wykryć w niniejszej pracy. Wówczas można byłoby podjąć próbę stworzenia metody pozwalającej na identyfikację takich grup. W dalszej kolejności można byłoby badać różne metody wyłaniania zwycięskiego komitetu projektów pod kątem uwzględniania głosów na takie spójne grupy – w szczególności kwestii dyskryminowania tychże grup w procesie wyborczym.

Bibliografia

[1] Pabulib files format.

[2] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[3] A. Grygorian and I. Iacob. A concise proof of the triangle inequality for the Jaccard distance.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] E. Schubert, S. Hess, and K. Morik. The Relationship of DBSCAN to Matrix Factorization and Spectral Clustering. *LWDA*, pages 330–334, 2018.

[6] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and Xi. Xu. DBSCAN Revisited, revisited. *ACM Transactions on Database Systems*, 42(3):1–21, 2017.

[7] J. Shi and J. Malik. Normalized cuts and image segmentation. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

[8] S. Szufa, P. Faliszewski, Ł. Janeczko, M. Lackner, A. Slinko, K. Sornat, and N. Talmon. How to sample approval elections?, Jul 2022.

[9] H. Zare, P. Shooshtari, A. Gupta, and R. R Brinkman. Data reduction for spectral clustering to analyze high throughput flow cytometry data. *BMC Bioinformatics*, 11(1), 2010.