

SSI
dokumentacja projektu semestralnego

Paweł Pietraszko

Konrad Matuszewski

grupa III H

28 maja 2020

Spis treści

1	Część I	2
1.1	Opis działania programu	2
1.2	Instrukcja obsługi	2
1.3	Dodatkowe informacje	3
2	Część II	4
2.1	Opis działania	4
2.2	Zastosowane wzory	4
2.3	Eksperymenty	5
2.4	Poprawki i rozwój programu	5
3	Pełen kod aplikacji	6
3.1	Day.cs	6
3.2	Provide.cs	7
3.3	Program.cs	12

1 Część I

1.1 Opis działania programu

Program ma za zadanie pokazać różnicę w skuteczności przypisania obiektu do grupy za pomocą poszczególnych algorytmów, tj. KNN, WKNN oraz metody Bayes’a.

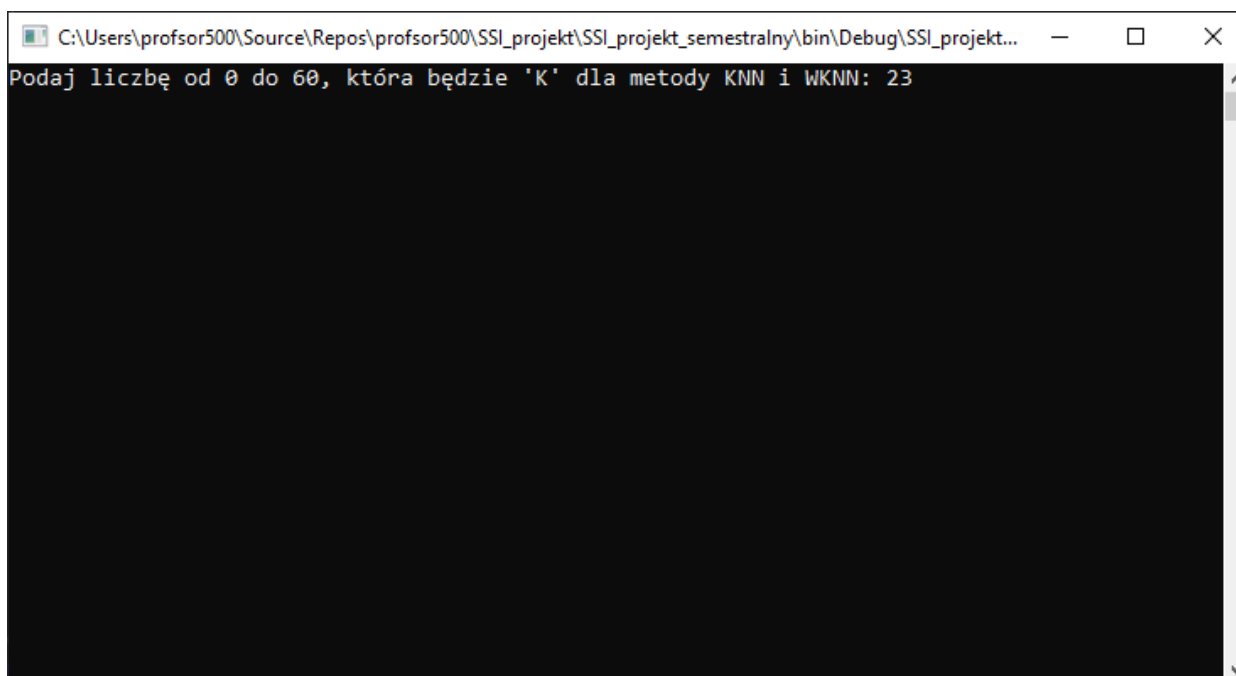
Dane użyte w programie opisują różne czynniki pogodowe, m. in. temperaturę, ilość opadów czy zachmurzenie. Zostały one pobrane z pewnej strony internetowej zawierającej informacje o pogodzie i gromadzącą takie dane z przeszłości, a następnie odpowiednio sformatowane, aby program mógł w prosty sposób je pobrać.

Program pobiera z pliku tekstowego WeatherDataSet.txt odpowiednie dane, które służą za zestaw uczący dla algorytmów, a następnie z pliku TestWeDatatSet.txt wczytuje inne obiekty, które porównywane są z tymi z zestawu uczącego, a następnie na konsoli wyświetlane jest podsumowanie oraz na przykładzie jednego z dni nieco bardziej

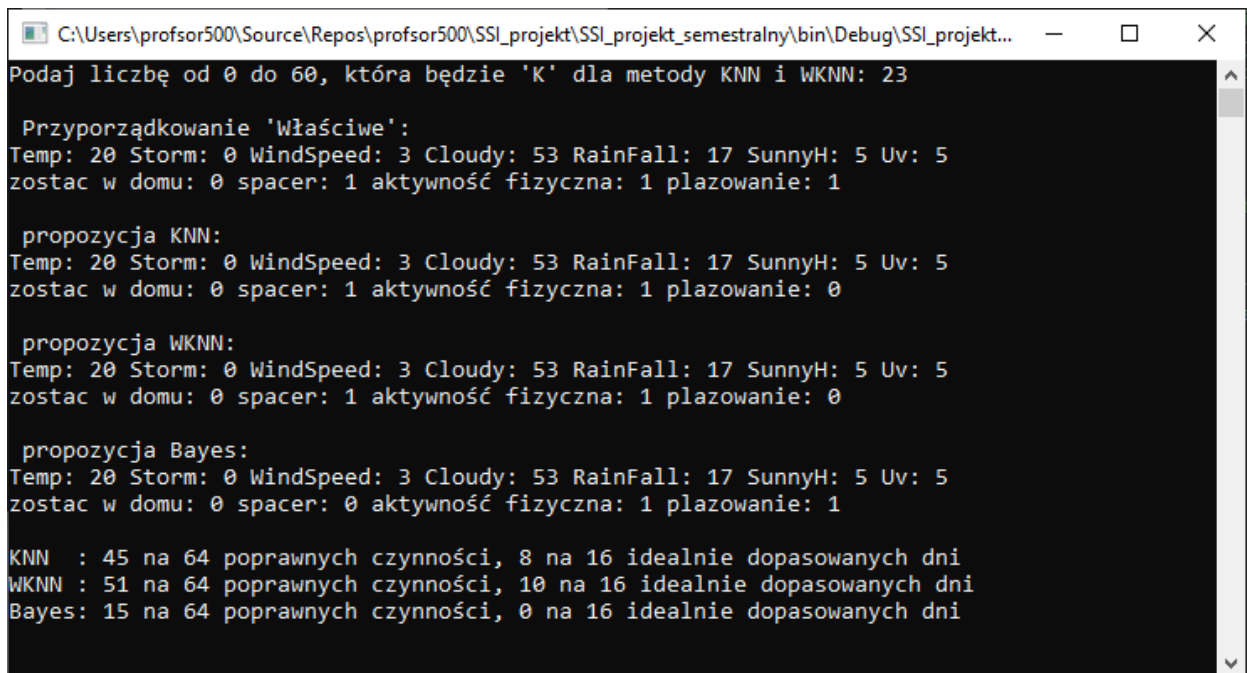
1.2 Instrukcja obsługi

Program nie został wyposażony w żadne menu, a jedynych modyfikacji jego działania można dokonać zmieniając kod źródłowy lub modyfikując w odpowiedni sposób pliki z danymi poprzez np dodanie nowych obiektów, modyfikacje aktualnych bądź usunięcie obecnych.

Po odpaleniu programu wyświetla nam się konsola z prośbą o podanie czynnika ‘k’ dla metody KNN i WKNN, czyli ilości najbliższych obiektów branych pod uwagę przy głosowaniu.



Po podaniu odpowiedniej liczby z przedziału od 1 do maksymalnie ilości obiektów w bazie (tutaj 60) zostanie wyświetlony odpowiedni wynik. Jeden z przykładowych dni zostanie wyświetlony bardziej szczegółowo oraz zostanie wyświetlone podsumowanie na temat skuteczności dla każdej z metod.



```
C:\Users\profsor500\Source\Repos\profsor500\SSL_projekt\SSL_projekt_semestralny\bin\Debug\SSL_projekt...
Podaj liczbę od 0 do 60, która będzie 'K' dla metody KNN i WKNN: 23

Przyporządkowanie 'Właściwe':
Temp: 20 Storm: 0 WindSpeed: 3 Cloudy: 53 RainFall: 17 SunnyH: 5 Uv: 5
zostac w domu: 0 spacer: 1 aktywność fizyczna: 1 plazowanie: 1

propozycja KNN:
Temp: 20 Storm: 0 WindSpeed: 3 Cloudy: 53 RainFall: 17 SunnyH: 5 Uv: 5
zostac w domu: 0 spacer: 1 aktywność fizyczna: 1 plazowanie: 0

propozycja WKNN:
Temp: 20 Storm: 0 WindSpeed: 3 Cloudy: 53 RainFall: 17 SunnyH: 5 Uv: 5
zostac w domu: 0 spacer: 1 aktywność fizyczna: 1 plazowanie: 0

propozycja Bayes:
Temp: 20 Storm: 0 WindSpeed: 3 Cloudy: 53 RainFall: 17 SunnyH: 5 Uv: 5
zostac w domu: 0 spacer: 0 aktywność fizyczna: 1 plazowanie: 1

KNN : 45 na 64 poprawnych czynności, 8 na 16 idealnie dopasowanych dni
WKNN : 51 na 64 poprawnych czynności, 10 na 16 idealnie dopasowanych dni
Bayes: 15 na 64 poprawnych czynności, 0 na 16 idealnie dopasowanych dni
```

1.3 Dodatkowe informacje

Program pisany był w języku C# w środowisku Visual Studio 2019 na komputerze z systemem operacyjnym Windows 10 w wersji 10.0.18363.836. Program powinien działać poprawnie na innych SO z rodziny windows oraz kod źródłowy dać się kompilować w innych środowiskach obsługujących C# pod warunkiem posiadania odpowiednich bibliotek.

2 Część II

2.1 Opis działania

Po odpaleniu programu pobierane są dane z pliku WeatherDataSet.txt i na ich podstawie dodane są obiekty Day do listy obiektów w klasie Provide.

Dla każdego dnia przyporządkowane są zalecane czynności za pomocą funkcji AdjustProposition() w formie słownika <czynność, 1,0>.

Następnie zostaje wczytany plik TestWeDatatSet.txt. Każda linijka jest zamieniana na obiekt dzień, a później zostają dopasowane czynności na podstawie metod KNN, WKNN, Bayes'a i porównane są z tymi właściwymi czynnościami, a wyniki efektywności są zliczone.

Po sprawdzeniu każdego dnia wypisane jest podsumowanie, tzn ile pojedynczych czynności oraz ile całych dni udało się dopasować której metodzie.

2.2 Zastosowane wzory

W KNN i WKNN odległość jest liczona za pomocą sumy pierwiastka kwadratu różnicy pomnożonej przez specjalny modyfikator poszczególnych wartości, tj:

$$x \in X, y \in Y, m \in M$$

$$d = \text{sum}i = 1\sqrt{((x_i - y_i)m_i)^2}$$

Gdzie:

X - pierwszy obiekt

x_i - i'ta dana obiektu X

Y -pierwszy obiekt

y_i - i'ta dana obiektu Y

M - zbiór modyfikatorów

m_i - i'ty modyfikator.

Zastosowanie modyfikatora ma na celu zrównoważenie wpływu poszczególnych danych poprzez zwiększenie wpływu niektórych z nich na odległość. (przykładowo: druga dana obiektu to informacja {0,1}, a piąta przyjmuje wartość od 0 do prawie 160).

W WKNN każdy wartość danego głosu jest tym ważniejsza, im bliżej naszego obiektu się on znajduje. Do tego celu zastosowany został następujący wzór:

$$\sqrt{1 - \sqrt{\frac{d_i}{d_{max}}}}$$

Gdzie:

d_i - dystans między dopasowanym a i'tym obiektem,

d_{max} - najdłuższy z dystansów branych pod uwagę.

Metoda Bayes'a korzysta z następującego wzoru:

$$\text{Max}(P(C_j) \prod_{i=1} P(x_i|C_j) : j \in J)$$

Gdzie:

$P(C_j)$ - prawdopodobieństwo zdarzenia C_j ,

$P(x_i|C_j)$ - prawdopodobieństwo zdarzenia x_i w momencie zajścia C_j ,

J - Wszystkie możliwe do wywnioskowania zdarzenia.

2.3 Eksperymenty

Nasz program pokazał, że metoda KNN i WKNN są dużo skuteczniejsze (w przypadku naszego programu), niż metoda Bayes'a, która przyporządkowywała odpowiedź z bardzo kiepskim skutkiem. Na każdy z szesnastu dni, dla którego miała przewidzieć po 4 wyniki 0,1, udało się przewidzieć zaledwie 15/64 (23,4%) czynności i ani jednego pełnego dnia spośród 16.

Metody KNN i WKNN okazały się o wiele skuteczniejsze. Niekiedy ich wyniki były równe, a czasami to WKNN okazywał się skuteczniejszy. Przykładowo dla $k=10$ KNN dopasował dobrze 57/64 (89%), z czego 12 na 16 całych dni, a WKNN 58/64 (90,5%) czynności, z czego 13 na 16 całych dni

2.4 Poprawki i rozwój programu

Jedną z możliwości rozwoju tego programu jest na pewno dodanie nowych algorytmów do porównania, jak sieć neuronowa chociażby lub zwiększenie wydajności lub dokładności już użytych, np. zastosowanie lepszego wzoru na wagę w WKNN.

Drugą możliwością jest poprawa komunikacji z użytkownikiem i dodanie przyjaznego mniej zaawansowanym użytkownikom GUI, które pozwoli w intuicyjniejszy i prostszy sposób wykonywać poszczególne operacje na algorytmach.

Na myśl przychodzi również połączenie programu z bazą danych, zamiast trzymać dane w plikach txt, co poprawiłoby na pewno ich bezpieczeństwo.

3 Pełen kod aplikacji

3.1 Day.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SSI_projekt_semestralny
8  {
9      class Day
10     {
11         public IDictionary<string, double> WeatherConditions { get; set; }
12         public IDictionary<string, int> Proposition { get; set; }
13         public Day(double temp, double storm, double windspeed, double cloudy,
14             double rainfall, double h, double uv)
15         {
16             WeatherConditions = new Dictionary<string, double>()
17             {
18                 {"Temp", temp },
19                 {"Storm", storm},
20                 {"WindSpeed", windspeed},
21                 {"Cloudy", cloudy},
22                 {"RainFall", rainfall},
23                 {"SunnyH", h},
24                 {"Uv", uv},
25             };
26             / obiekt.AdjustProposition()
27             Proposition = new Dictionary<string, int>()
28             {
29                 {"zostac w domu",0 },
30                 {"spacer",0 },
31                 {"aktywnosc fizyczna",0 },
32                 {"plazowanie",0}
33             };
34         }
35         public void SetProposition(IDictionary<string, int> prop)
36         {
37             this.Proposition = prop;
38         }
39         public double DistanceToOther(Day other)
40         {
41             double Result = 0;
42             double[] modifiers = new double[] {5,20,7,1,1,4,8};
43             int i = 0;
44             foreach (var key in this.WeatherConditions.Keys)
45             {
46                 Result += Math.Sqrt(Math.Pow((this.WeatherConditions[key] -
47                     other.WeatherConditions[key])*modifiers[i], 2));
48                 i++;
49             }
50             return Result;
51         }
52         public string toString()
53         {
54             string result = "";
55             foreach(var item in WeatherConditions) result+=item.Key+": "+ item.
56                 Value.ToString()+" ";
57             result += "\n";
```

```

56         foreach (var item in Proposition) result += item.Key + ": " + item.
           Value.ToString() + " ";
57     return result;
58
59 }
60
61 public void AdjustProposition()
62 {
63     Proposition["zostac w domu"] = StayHome();
64     Proposition["spacer"] = Walk();
65     Proposition["aktywnosc fizyczna"] = Workout();
66     Proposition["plazowanie"] = Beaching();
67 }
68 int StayHome()
69 {
70     if (this.WeatherConditions["Storm"] == 1) return 1;
71     else if (this.WeatherConditions["Uv"] >= 8) return 1;
72     else if (this.WeatherConditions["RainFall"] > 70) return 1;
73     return 0;
74 }
75 int Walk()
76 {
77     if ((this.WeatherConditions["Storm"] == 1 || this.WeatherConditions
78         ["RainFall"] > 70) && this.WeatherConditions["SunnyH"] <= 3 )
79         return 0;
80     else if (this.WeatherConditions["Uv"] >= 8 && this.
81         WeatherConditions["Cloudy"] <70) return 0;
82     else if (this.WeatherConditions["RainFall"] > 70 && this.
83         WeatherConditions["SunnyH"] <= 3) return 0;
84     return 1;
85 }
86 int Workout()
87 {
88     if (this.WeatherConditions["Storm"] == 1 && this.WeatherConditions[
89         "SunnyH"] <= 3) return 0;
90     else if (this.WeatherConditions["Uv"] >= 8 && this.
91         WeatherConditions["SunnyH"] > 8) return 0;
92     else if (this.WeatherConditions["RainFall"] > 70 && this.
93         WeatherConditions["SunnyH"] <= 3) return 0;
94     else if (this.WeatherConditions["Temp"] > 25) return 0;
95     return 1;
96 }
97 int Beaching()
98 {
99     if (this.WeatherConditions["Storm"] == 1 || this.WeatherConditions[
100         "SunnyH"] <= 3) return 0;
101     else if (this.WeatherConditions["Uv"] >= 8 || this.
102         WeatherConditions["Uv"] <= 4) return 0;
103     else if (this.WeatherConditions["RainFall"] > 70 && this.
104         WeatherConditions["SunnyH"] <= 3) return 0;
105     else if (this.WeatherConditions["Temp"] < 20) return 0;
106     return 1;
107 }
108 }
109
110 }

```

3.2 Provide.cs

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;

```

```

5 using System.Threading.Tasks;
6 using System.IO;
7 using System.Runtime.CompilerServices;
8
9 namespace SSI_projekt_semestralny
10 {
11     class Provide
12     {
13         List<Day> DaysList { get; set; }
14         public Provide()
15         {
16             DaysList = new List<Day>();
17         }
18         public int DaysCount()
19         {
20             return DaysList.Count;
21         }
22         public void MultiMethodsComparison(int k, string pathtodataset)
23         {
24             var CorrectKNN = new int[] { 0, 0 };
25             var CorrectWKNN = new int[] { 0, 0 };
26             var CorrectBayes = new int[] { 0, 0 };
27             var lines = File.ReadAllLines( @pathtodataset);
28             Day dzien;
29             foreach (var line in lines.Skip(1).ToArray())
30             {
31                 string l = line.Replace("\t", "");
32                 double[] double_line = Array.ConvertAll(l.Split(';'), Double.
33                     Parse);
34                 if (double_line.Length == 7)
35                 {
36                     dzien = new Day(double_line[0], double_line[1], double_line
37                         [2], double_line[3], double_line[4], double_line[5],
38                         double_line[6]);
39                     dzien.AdjustProposition();
40                     var wkndict = WKNN(dzien, k);
41                     var knndict = KNN(dzien, k);
42                     var bayesdict = Bayes(dzien);
43                     if (dzien.Proposition.Count == wkndict.Count && !dzien.
44                         Proposition.Except(wkndict).Any()) { CorrectWKNN[0] +=
45                         4; CorrectWKNN[1] += 1; }
46                     else { foreach (var key in wkndict.Keys) if (wkndict[key]
47                         == dzien.Proposition[key]) CorrectWKNN[0]++; }
48
49                     if (dzien.Proposition.Count == knndict.Count && !dzien.
50                         Proposition.Except(knndict).Any()) { CorrectKNN[0] +=
51                         4; CorrectKNN[1] += 1; }
52                     else { foreach (var key in knndict.Keys) if (knndict[key]
53                         == dzien.Proposition[key]) CorrectKNN[0]++; }
54
55                     if (dzien.Proposition.Count == bayesdict.Count && !dzien.
56                         Proposition.Except(bayesdict).Any()) { CorrectBayes[0]
57                         += 4; CorrectBayes[1] += 1; }
58                     else { foreach (var key in bayesdict.Keys) if (bayesdict[
59                         key] == dzien.Proposition[key]) CorrectBayes[0]++; }
60                 }
61             }
62             Console.WriteLine("KNN : {0} na {1} poprawnych czynnosci, {2} na
63                 {3} idealnie dopasowanych dni", CorrectKNN[0], (lines.Length-1)
64                 * 4, CorrectKNN[1], (lines.Length - 1));
65             Console.WriteLine("WKNN : {0} na {1} poprawnych czynnosci, {2} na
66                 {3} idealnie dopasowanych dni", CorrectWKNN[0], (lines.Length -

```



```

1) * 4, CorrectWKNN[1], (lines.Length - 1));
53 Console.WriteLine("Bayes: {0} na {1} poprawnych czynnosci, {2} na
    {3} idealnie dopasowanych dni", CorrectBayes[0], (lines.Length
    - 1) * 4, CorrectBayes[1], (lines.Length - 1));
54 }
55
56 public IDictionary<string, int> Bayes(Day dzien)
57 {
58     var ResultPropositions = new Dictionary<string, int>();
59
60     var TempLims = SetLims(dzien.WeatherConditions["Temp"], 5);
61     var WindSpeedLims = SetLims(dzien.WeatherConditions["WindSpeed"],
62     3);
63     var CloudyLims = SetLims(dzien.WeatherConditions["Cloudy"], 25);
64     var RainFallLims = SetLims(dzien.WeatherConditions["RainFall"], 15);
65     ;
66     var SunnyHLims = SetLims(dzien.WeatherConditions["SunnyH"], 3);
67     var UvLims = SetLims(dzien.WeatherConditions["Uv"], 3);
68     foreach (var key in dzien.Proposition.Keys)
69     {
70         int[] DecCount = {0 ,0};
71         int[] TempCount = { 0, 0 };
72         int[] StormCount = { 0, 0 };
73         int[] WindCount = { 0, 0 };
74         int[] CloudyCount = { 0, 0 };
75         int[] RainFallCount = { 0, 0 };
76         int[] SunnyHCount = { 0, 0 };
77         int[] UvCount = { 0, 0 };
78         double[] result = { 0, 0 };
79         foreach (var day in DaysList)
80         {
81             if (day.Proposition[key] == 1)
82             {
83                 DecCount[1]++;
84                 if (day.WeatherConditions["Temp"] >= TempLims[0] && day
85                     .WeatherConditions["Temp"] < TempLims[1]) TempCount
86                     [1]++;
87                 if (day.WeatherConditions["Storm"] == dzien.
88                     WeatherConditions["Storm"]) StormCount[1]++;
89                 if (day.WeatherConditions["WindSpeed"] >= WindSpeedLims
90                     [0] && day.WeatherConditions["WindSpeed"] <
91                     WindSpeedLims[1]) WindCount[1]++;
92                 if (day.WeatherConditions["Cloudy"] >= CloudyLims[0] &&
93                     day.WeatherConditions["Cloudy"] < CloudyLims[1])
94                     CloudyCount[1]++;
95                 if (day.WeatherConditions["RainFall"] >= RainFallLims
96                     [0] && day.WeatherConditions["RainFall"] <
97                     RainFallLims[1]) RainFallCount[1]++;
98                 if (day.WeatherConditions["SunnyH"] >= SunnyHLims[0] &&
99                     day.WeatherConditions["SunnyH"] < SunnyHLims[1])
100                     SunnyHCount[1]++;
101                 if (day.WeatherConditions["Uv"] >= UvLims[0] && day.
102                     WeatherConditions["Uv"] < UvLims[1]) UvCount[1]++;
103             }
104             else //czyli jezeli Day.Proposition[key]==0
105             {
106                 DecCount[0]++;
107                 if (day.WeatherConditions["Temp"] >= TempLims[0] && day
108                     .WeatherConditions["Temp"] < TempLims[1]) TempCount
109                     [0]++;
110                 if (day.WeatherConditions["Storm"] == dzien.
111                     WeatherConditions["Storm"]) StormCount[1]++;
112                 if (day.WeatherConditions["WindSpeed"] >= WindSpeedLims

```

```

        [0] && day.WeatherConditions["WindSpeed"] <
        WindSpeedLims[1]) WindCount[0]++;
96     if (day.WeatherConditions["Cloudy"] >= CloudyLims[0] &&
        day.WeatherConditions["Cloudy"] < CloudyLims[1])
        CloudyCount[0]++;
97     if (day.WeatherConditions["RainFall"] >= RainFallLims
        [0] && day.WeatherConditions["RainFall"] <
        RainFallLims[1]) RainFallCount[0]++;
98     if (day.WeatherConditions["SunnyH"] >= SunnyHLims[0] &&
        day.WeatherConditions["SunnyH"] < SunnyHLims[1])
        SunnyHCount[0]++;
99     if (day.WeatherConditions["Uv"] >= UvLims[0] && day.
        WeatherConditions["Uv"] < UvLims[1]) UvCount[0]++;
100 }
101
102 }
103 result[1] = DecCount[1] * TempCount[1] * StormCount[1] *
        WindCount[1] * CloudyCount[1] * RainFallCount[1] *
        SunnyHCount[1] * UvCount[1] / (DaysList.Count * Math.Pow(
        DecCount[1], 7));
104 result[0] = DecCount[0] * TempCount[0] * StormCount[0] *
        WindCount[0] * CloudyCount[0] * RainFallCount[0] *
        SunnyHCount[0] * UvCount[0] / (DaysList.Count * Math.Pow(
        DecCount[0], 7));
105 if (result[1] >= result[0]) ResultPropositions.Add(key, 1);
106 else ResultPropositions.Add(key, 0);
107
108 }
109 return ResultPropositions;
110 }
111
112 int[] SetLims(double value, int bar)
113 {
114     int UpLim = bar;
115     while (value >= UpLim) UpLim += bar;
116     return new int[] {UpLim-bar, UpLim};
117 }
118
119 public void GetDays(string path)
120 {
121     int WrongLine = 0;
122     var lines = File.ReadAllLines(@path);
123     foreach (var line in lines.Skip(1).ToArray()) {
124         string l = line.Replace("\t", "");
125         double[] double_line = Array.ConvertAll(l.Split(';'), Double.
            Parse);
126         if (double_line.Length == 7)
127         {
128             DaysList.Add(new Day(double_line[0], double_line[1],
                double_line[2], double_line[3], double_line[4],
                double_line[5], double_line[6]));
129             DaysList[DaysList.Count - 1].AdjustProposition();
130         }
131         else { WrongLine++; }
132     }
133     if(WrongLine>0) Console.WriteLine("nie udalo sie przekonwertowac
        {0} na {1} wszystkich wyrazow", WrongLine, lines.Length-1);
134 }
135
136 public void GetDays(string path, int number)
137 {
138     int WrongLine = 0;
139     var lines = File.ReadAllLines(@path);

```

```

140         for (int i=1;i<number;i++)
141         {
142             double[] double_line = Array.ConvertAll(lines[i].Split(';'),
143                 Double.Parse);
144             if (double_line.Length >= 7)
145             {
146                 DaysList.Add(new Day(double_line[0], double_line[1],
147                     double_line[2], double_line[3], double_line[4],
148                     double_line[5], double_line[6]));
149                 DaysList[DaysList.Count - 1].AdjustProposition();
150             }
151             else WrongLine++;
152         }
153         if (WrongLine > 0) Console.WriteLine("nie udalo sie przekonwertowac
154             {0} na {1} wszystkich wyrazow", WrongLine, lines.Length - 1);
155     }
156
157     public void PrintList()
158     {
159         foreach (Day d in DaysList) Console.WriteLine(d.toString()+"\n");
160     }
161
162     public void PrintList(int x)
163     {
164         if (DaysList.Count<x)
165             for (int i=0;i< DaysList.Count; i++) Console.WriteLine(DaysList
166                 [i].toString() + "\n");
167         else
168             for (int i = 0; i < x; i++) Console.WriteLine(DaysList[i].
169                 toString() + "\n");
170     }
171
172     public IDictionary<string, int> KNN(Day dzien, int k)
173     {
174         List<knnStruct> distances = new List<knnStruct>();
175         for (int i = 0; i < DaysList.Count; i++) distances.Add(new
176             knnStruct(DaysList[i], dzien.DistanceToOther(DaysList[i])));
177         distances = distances.OrderBy(x => x.distance).ToList();
178         var kdistances = distances.Take(k).ToList();
179         var propos = new List<IDictionary<string, int>>();
180         var proposCount = new List<int>();
181
182         for (int i = 0; i < kdistances.Count; i++)
183         {
184             bool nieznalesziono = true;
185             for (int j = 0; j < propos.Count; j++)
186             {
187                 if (propos[j].Count == kdistances[i].day.Proposition.
188                     Count && !propos[j].Except(kdistances[i].day.
189                         Proposition).Any()) { proposCount[j]++;
190                     nieznalesziono = false; break; };
191             }
192             if (nieznalesziono)
193             {
194                 propos.Add(kdistances[i].day.Proposition);
195                 proposCount.Add(1);
196             }
197         }
198     }

```

```

193         return propos[proposCount.IndexOf(proposCount.Max())];
194     }
195     public IDictionary<string, int> WKNN(Day dzien, int k)
196     {
197         List<knnStruct> distances = new List<knnStruct>();
198         for (int i = 0; i < DaysList.Count; i++) distances.Add(new
199             knnStruct(DaysList[i], dzien.DistanceToOther(DaysList[i])));
200         distances = distances.OrderBy(x => x.distance).ToList();
201         var kdistances = distances.Take(k).ToList();
202         double maxd = kdistances.Max(x => x.distance);
203         var propos = new List<IDictionary<string, int>>();
204         var proposCount = new List<double>();
205
206         for (int i = 0; i < kdistances.Count; i++)
207         {
208             bool nieznaleziono = true;
209             for (int j = 0; j < propos.Count; j++)
210             {
211                 if (propos[j].Count == kdistances[i].day.Proposition.Count
212                     && !propos[j].Except(kdistances[i].day.Proposition).Any
213                     ()) { proposCount[j] += Math.Sqrt(1 - Math.Sqrt(
214                         kdistances[i].distance / maxd)); nieznaleziono = false;
215                         break; };
216             }
217             if (nieznaleziono)
218             {
219                 propos.Add(kdistances[i].day.Proposition);
220                 proposCount.Add(Math.Sqrt(1 - Math.Sqrt(kdistances[i].
221                     distance / maxd)));
222             }
223         }
224         return propos[proposCount.IndexOf(proposCount.Max())];
225     }
226     struct knnStruct
227     {
228         public Day day;
229         public double distance;
230         public knnStruct(Day day, double distance)
231         {
232             this.day = day;
233             this.distance = distance;
234         }
235     }
236 }

```

3.3 Program.cs

```

1 using System;
2 using System.Collections.Generic;
3 using System.Diagnostics.CodeAnalysis;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace SSI_projekt_semestralny
9 {
10     class Program
11     {
12         static void Main(string[] args)
13         {

```

```

14     var przewidywanie = new Provide();
15     przewidywanie.GetDays(@"C:\Users\profsor500\Desktop\Studia\
        SystemySztucznejInteligencji\SSI_projekt_semestralny\
        SSI_projekt_semestralny\WeatherDataSet.txt");
16     int number;
17     string k;
18     do
19     {
20         Console.Clear();
21         Console.Write("Podaj liczbe od 0 do {0}, ktora bedzie 'K' dla
            metody KNN i WKNN: ", przewidywanie.DaysCount()); k =
            Console.ReadLine();
22     } while (!Int32.TryParse(k, out number) || number <= 0 || number >
        przewidywanie.DaysCount());
23     Day dzien0 = new Day(20, 0, 3, 53, 17, 5, 5);
24     Day dzien1 = new Day(20, 0, 3, 53, 17, 5, 5);
25     Day dzien2 = new Day(20, 0, 3, 53, 17, 5, 5);
26     Day dzien3 = new Day(20, 0, 3, 53, 17, 5, 5);
27     dzien0.SetProposition(przewidywanie.KNN(dzien0, number));
28     dzien1.SetProposition(przewidywanie.WKNN(dzien1, number));
29     dzien2.SetProposition(przewidywanie.Bayes(dzien2));
30     dzien3.AdjustProposition();
31     Console.WriteLine("\n Przyporządkowanie 'Wlasciwe':");
32     Console.WriteLine(dzien3.toString());
33     Console.WriteLine("\n propozycja KNN:");
34     Console.WriteLine(dzien0.toString());
35     Console.WriteLine("\n propozycja WKNN:");
36     Console.WriteLine(dzien1.toString());
37     Console.WriteLine("\n propozycja Bayes:");
38     Console.WriteLine(dzien2.toString());
39     Console.WriteLine();
40     przewidywanie.MultiMethodsComparison(number, @"C:\Users\profsor500\
        Desktop\Studia\SystemySztucznejInteligencji\
        SSI_projekt_semestralny\SSI_projekt_semestralny\TestWeDatatSet.
        txt");
41     Console.ReadKey();
42     }
43 }
44 }

```
