Politechnika Wrocławska
Wydział Elektroniki

Metody techniki systemów w medycynie 2
Projekt

Autorzy:
Aleksandra Ziobrowska 241129
Mariusz Wiśniewski 241393

Prowadzący:
mgr inż. Paweł Zyblewski

18 listopada 2020

Spis treści

1 Wybór tematu

Wybranym przez nas tematem jest **"Komputerowe wspomaganie diagnozowania zawałów z wykorzystaniem algorytmu k-NN"**. W drugim etapie projektu zaprezentowaliśmy opis problemu medycznego jako zadania klasyfikacji oraz wyznaczenie rankingu cech pod względem ich przydatności do klasyfikacji. W kolejnych etapach należało zaplanować i zrealizować badania eksperymentalne, a następnie przeanalizować otrzymane wyniki.

2 Opis problemu medycznego jako zadania klasyfikacji

Podejmowanym przez nas problemem medycznym jest wykorzystanie metod sztucznej inteligencji (algorytmu k-NN) do wspomagania diagnozowania zawałów serca u pacjentów. Do stworzenia rozwiązania będą wykorzystane dane treningowe dotyczące profilu medycznego pacjenta. Program na podstawie danych będzie wiązać symptomy oraz profil pacjenta z chorobą serca lub informować o braku możliwości powiązania objawów z jego przyczyną, co powoduje, że podejmowany problem medyczny jest problemem klasyfikacji wieloklasowej.

Klasyfikowane będą choroby związane z sercem na podstawie zawartych w danych treningowych cech, dotyczących historii medycznej oraz stanu zdrowotnego pacjenta. Zagregowany zbiór danych zawiera pięć klas oraz łącznie 901 rekordów. W ramach wstępnego przetwarzania zbioru danych, niektóre cechy zostaną pominięte, co zostało opisane w sekcji 3. W ramach optymalizacji stworzonego modelu zostanie zastosowany tuning hiperparametrów dla parametrów charakterystycznych dla klasyfikatora k-NN.

2.1 Liczba klas i ich medyczny sens

Poniżej przedstawione zostały klasy (diagnozy) wraz z ich licznościami - liczbą pacjentów. Wszystkie klasy są powiązane z chorobami serca.

1. Ból pochodzenia innego niż serce (Pain of non-heart origin) - 230
2. Dławica piersiowa (Angina pectoris) - 142
3. Dławica naczynioskurczowa (Angina pectoris - Prinzmetal variant) - 68
4. Zawał serca pełnościenny (Myocardial infraction (transmural)) - 263
5. Zawał serca podwsierdziowy (Myocardial infraction (subendocardial)) - 198

2.2 Liczba cech i ich charakterystyka

Każdy obiekt uczący (pacjent) jest charakteryzowany przez 59 cech. Stany cechy dyskretnej przybierają tylko niektóre wartości należące do określonego przedziału liczbowego. Cechom ciągłym są przyporządkowane nieskończone lub nieprzeliczalne zbiory stanów. Wiek w rozumieniu ilości skończonych lat jest również cechą dyskretną.

Tablica 1: Cechy obiektu uczącego

Nr	Cecha	Typ	Nr	Cecha	Typ
1	Wiek	Dyskretne	31	Nitraty	Dyskretne
2	Płeć	Dyskretne	32	Beta-blokery	Dyskretne
3	Lokalizacja bólu	Dyskretne	33	Naparstnica	Dyskretne
4	Promieniowanie bólu w klatce piersiowej:	Dyskretne	34	Niesteroidowe działanie przeciwzapalne	Dyskretne
5	Charakter bólu	Dyskretne	35	Leki zobojętniające / blokery H2	Dyskretne
6	Kontekst bólu	Dyskretne	36	Skurczowe ciśnienie krwi (liczba całkowita)	Dyskretne
7	Liczba godzin od wystąpienia bólu	Dyskretne	37	Rozkurczowe ciśnienie krwi (liczba całkowita)	Dyskretne
8	Czas trwania ostatniego epizodu	Dyskretne	38	Tętno (liczba całkowita)	Dyskretne
9	Nudności	Dyskretne	39	Częstość oddechów (liczba całkowita)	Dyskretne
10	Potliwość	Dyskretne	40	Szmer oddechowy	Dyskretne
11	Kołatanie serca	Dyskretne	41	Sinica	Dyskretne
12	Duszność	Dyskretne	42	Bladość	Dyskretne
13	Zawroty głowy / omdlenia	Dyskretne	43	Szmer skurczowy	Dyskretne
14	Odbijanie	Dyskretne	44	Szmer rozkurczowy	Dyskretne
15	Czynniki paliatywne	Dyskretne	45	Obrzęk	Dyskretne
16	Wcześniejszy ból tego typu w klatce piersiowej	Dyskretne	46	Ton serca typu S3	Dyskretne
17	Konsultacja z lekarzem w sprawie wcześniejszego bólu	Dyskretne	47	Ton serca typu S4	Dyskretne
18	Wcześniejszy ból związany z sercem	Dyskretne	48	Wrażliwość ściany klatki piersiowej	Dyskretne
19	Wcześniejszy ból spowodowany zawałem serca	Dyskretne	49	Nadmierna potęga	Dyskretne
20	Wcześniejszy ból spowodowany chorobą niedokrwienną serca	Dyskretne	50	Nowa fala Q.	Dyskretne
21	Wcześniejszy zawał serca	Dyskretne	51	Dowolna fala Q.	Dyskretne
22	Wcześniejsza choroba niedokrwienna serca	Dyskretne	52	Nowa elewacja segmentu ST	Dyskretne
23	Wcześniej nietypowy ból w klatce piersiowej	Dyskretne	53	Dowolne podniesienie segmentu ST	Dyskretne
24	Zastoinowa niewydolność serca	Dyskretne	54	Nowe obniżenie odcinka ST	Dyskretne
25	Choroba naczyń obwodowych	Dyskretne	55	Dowolne obniżenie odcinka ST	Dyskretne
26	Przepuklina rozworu przełykowego	Dyskretne	56	Nowa inwersja załamka T.	Dyskretne
27	Nadciśnienie	Dyskretne	57	Dowolna inwersja załamka T.	Dyskretne
28	Cukrzyca	Dyskretne	58	Nowa wada przewodzenia śródkomorowego	Dyskretne
29	Palący	Dyskretne	59	Dowolna wada przewodzenia śródkomorowego	Dyskretne
30	Diuretyki	Dyskretne			

3 Wyznaczenie rankingu cech pod względem ich przydatności do klasyfikacji

Do stworzenia rankingu cech wykorzystaliśmy biblioteki *numpy*[4], *pandas*[5] oraz *scikit-learn*[6]. Biblioteka *numpy* pozwoliła nam złączyć załadowane pliki w jedną listę, *pandas* - służyła głównie do ładnej prezentacji danych, a *scikit-learn* umożliwiła wyznaczenie rankingu cech pod względem ich przydatności do klasyfikacji.

```
import numpy as np
import pandas as pd
from sklearn.feature_selection import chi2
```

Listing 1: Wykorzystywane biblioteki

Stworzyliśmy również własne funkcje, które odpowiadają za załadowanie danych z plików.

```
from data_parser import load_features_names, load_all_files, load_data,
    ↪ load_diseases_names
from data_summary import show_distribution
```

Listing 2: Własne funkcje przygotowujące dane

Poniżej wykorzystywane funkcje pozwoliły nam załadować wszystkie dane z plików. Funkcja *load_all_files()* pobierała informacje o cechach osobników z plików *inne.txt, ang_prect.txt, ang_prct_2.txt, mi.txt* oraz *mi_np.txt*. Funkcje *load_features_names()* oraz *load_diseases_names()* pobierały odpowiednio nazwy cech oraz chorób z pliku *MYOCARDIAL INFRACTION DIAGNOSIS*, którego rozszerzenie, dla wygodniejszego przetwarzania, zmieniliśmy ręcznie na *.txt*.

```
    datasets = load_all_files()
    features_names = load_features_names()
    diseases_names = load_diseases_names()
```

Listing 3: Pobranie danych z plików

Funkcja *load_data()* rozdziela załadowane pliki na listy po to, abyśmy byli w stanie wypisać liczebność każdej z klas przy użyciu *print_classes_strength()*. Następnie wyświetlamy diagram kołowy przedstawiający zależności między liczebnościami poszczególnych klas.

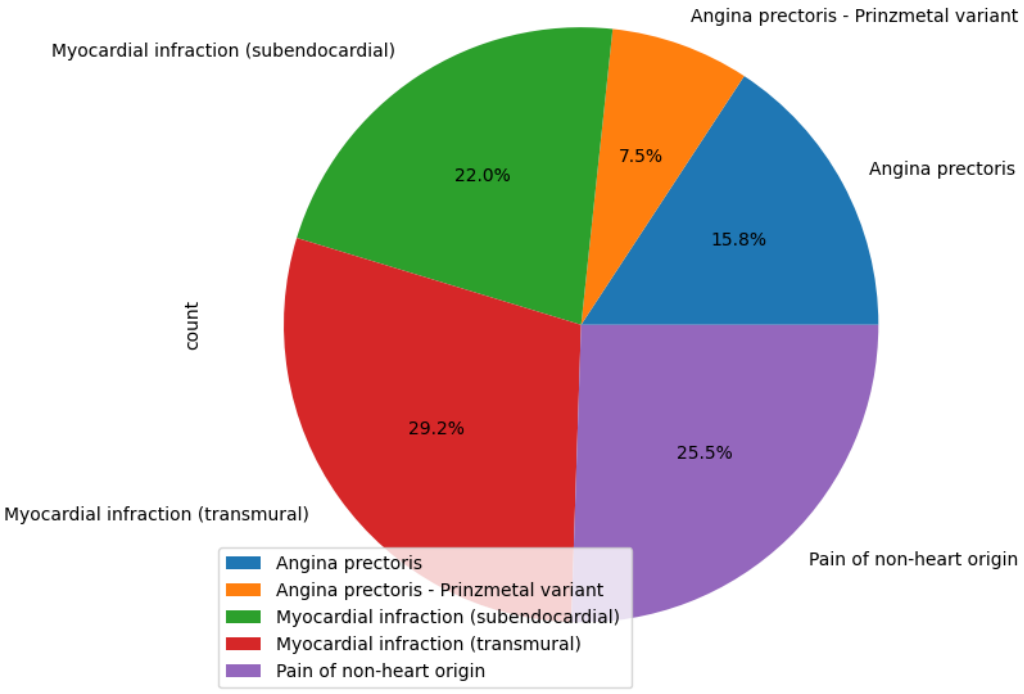
```
    data = load_data(datasets)
    data.print_classes_strength()

    show_distribution(datasets, features_names)
```

Listing 4: Wyświetlenie informacji o liczebności klas

```
                Pain of non heart origin: 230
                       Angina prectoris: 142
  Angina prectoris - Prinzmetal variant:  68
     Myocardial infraction (transmural): 263
 Myocardial infraction (subendocardial): 198
```

Listing 5: Liczebności klas



Rysunek 1: Diagram liczebności klas

Przygotowanie do wyznaczenia rankingu cech polegało na przygotowaniu dwóch list - *features*, będącą złączoną listą wszystkich cech poszczególnych osobników oraz *classes* zawierającą nazwy chorób, zachowując przy tym informacje o tym, której klasie (chorobie) odpowiadają konkretne wartości cech (wykorzystywany jest ten sam indeks).

```
1     features = np.concatenate([*datasets])
2     classes = np.concatenate(
3         [np.full(dataset.shape[0], diseases_names[index]) for (index, dataset)
  ↪   in enumerate(datasets)])
```

Listing 6: Przygotowanie danych do wyznaczenia rankingu

Ponieważ nasze dane mają charakter dyskretny oraz są cechami jakościowymi, co oznacza, że mogą one przyjąć ograniczony zakres wartości, zdecydowaliśmy się zastosować test *chi2*, który jest najbardziej odpowiednim dla tego typu danych.

Test *chi2* [9] [3] jest wykorzystywany w statystyce do sprawdzenia niezależności dwóch wydarzeń.

Wartość *chi2* wyrażona jest wzorem:

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i} \tag{1}$$

, gdzie c to liczba stopni swobody, O to zaobserwowana wartość, a E - oczekiwana wartość.

Przy tworzeniu rankingu cech chcemy zdeterminować związek pomiędzy niezależną cechą (np. płeć) oraz klasą (np. zawał serca). Jeśli parametry te są od siebie niezależne, to obserwowana wartość O jest bliska oczekiwanej wartości E, więc wartość *chi2* jest mała.

Hipoteza zerowa zakłada, że cecha i klasa są od siebie niezależne.

Im większa wartość *chi2*, tym większa zależność między cechą i klasą, więc cechę tą możemy uznać za bardziej istotną dla naszego problemu.

W dalszej części przedstawimy przyjęty przez nas poziom istotności, który pozwoli nam stwierdzić, dla których cech jesteśmy w stanie odrzucić hipotezę zerową, a tym samym założyć, że istnieje związek między cechą, a klasą.

```
    scores, p_values = chi2(features, classes)
```

Listing 7: Wywołanie funkcji generującej wartości wykorzystywane w rankingu cech

Po otrzymaniu wartości statystyki *chi2* oraz prawdopodobieństwa *p_value*, w celu ładnego wyświetlenia danych, korzystając z biblioteki *pandas*, utworzyliśmy jeden wspólny *DataFrame*, w którym dodaliśmy podpisy kolumn. Indeks został podniesiony o 1 po to, aby był on zgodny z indeksem występującym przy nazwie cechy w pliku *MYOCARDIAL INFRACTION DIAGNOSIS.doc*.

Następnie wyświetlamy otrzymane dane, zaokrąglając wartości do trzech miejsc dziesiętnych oraz sortując je malejąco po wartościach statystyki *chi2*.

```
    features_with_values = pd.concat(
        [pd.DataFrame(features_names, columns=['Features']), pd.DataFrame(
    ↪ scores, columns=['Scores']),
         pd.DataFrame(p_values, columns=['P_values'])], axis=1)
    features_with_values.index += 1  # To match with feature's number

    print(features_with_values.sort_values('Scores', ascending=False).round(3)
    ↪ )
```

Listing 8: Wyświetlenie wyników wygenerowanych przez funkcję *chi2*

Tablica 2: Wyniki wygenerowane przez funkcję chi2

	Features	Scores	P_values
36	Systolic blood pressure	1980.235	0.000
7	Number of hours since onset	978.578	0.000
3	Pain location	340.518	0.000
54	New ST segment depression	223.468	0.000

50	New Q wave	200.256	0.000
56	New T wave inversion	193.398	0.000
52	New ST segment elevation	188.224	0.000
55	Any ST segment depression	177.000	0.000
58	New intraventricular conduction defect	159.892	0.000
57	Any T wave inversion	151.672	0.000
39	Respiration rate	120.295	0.000
44	Diastolic murmur	117.165	0.000
59	Any intraventricular conduction defect	117.095	0.000
22	Prior angina prectoris	116.643	0.000
4	Chest pain radiation	114.724	0.000
38	Heart rate	109.556	0.000
51	Any Q wave	108.816	0.000
46	S3 gallop	105.087	0.000
18	Prior pain related to heart	101.630	0.000
19	Prior pain due to MI	89.169	0.000
47	S4 gallop	87.111	0.000
20	Prior pain due to angina prectoris	85.870	0.000
24	Congestive heart failure	85.266	0.000
43	Systolic murmur	84.497	0.000
26	Hiatal hernia	83.614	0.000
13	Dizziness/syncope	83.285	0.000
11	Palpitations	82.349	0.000
32	Beta blockers	79.733	0.000
37	Diastolic blood pressure	78.563	0.000
31	Nitrates	76.575	0.000
53	Any ST segment elevation	75.294	0.000
35	Antacids/H2 blockers	73.567	0.000
8	Duration of the last episode	63.415	0.000
33	Digitalis	60.113	0.000
41	Cyanosis	58.806	0.000
21	Prior MI	56.670	0.000
1	Age	56.543	0.000
40	Rales	54.042	0.000
48	Chest wall tenderness	53.812	0.000
5	Pain character	50.834	0.000
23	Prior atypical chest pain	49.971	0.000
9	Nausea	45.571	0.000
42	Pallor	45.247	0.000
14	Burping	43.215	0.000
27	Hypertension	41.360	0.000
16	Prior chest pain of this type	40.414	0.000
12	Dyspnea	36.725	0.000
34	Nonsteroidal anti-inflammatory	35.548	0.000
25	Peripheral vascular disease	28.190	0.000

49	Diaphoresis	27.678	0.000
6	Onset of pain	21.993	0.000
28	Diabetes	15.453	0.004
17	Physician consulted for prior pain	15.174	0.004
29	Smoker	15.063	0.005
45	Oedema	14.004	0.007
15	Palliative factors	12.859	0.012
2	Sex	10.844	0.028
10	Diaphoresis	6.584	0.160
30	Diuretics	5.190	0.268

Przyjęty przez nas poziom istotności *alpha* to 0.05 [2]. Wynika z tego, że nie odrzucamy hipotezy zerowej dla cech *Diaphoresis* oraz *Diuretics*, a tym samym utrzymujemy, że zależność cecha-klasa dla tych przypadków nie występuje. Przedstawione rozumowanie pozwala nam usunąć te dwie cechy z listy.

```
    alpha = 0.05
    for index, row in features_with_values.iterrows():
        p_value = row['P_values']
        if p_value > alpha:
            features_with_values.drop(index, inplace=True)

    print(features_with_values.sort_values('Scores', ascending=False).round(3)
    )
```

Listing 9: Wyświetlenie rankingu po redukcji cech

Ranking cech po opisanej wyżej modyfikacji został przedstawiony poniżej.

Tablica 3: Ranking po redukcji cech

	Features	Scores	P_values
36	Systolic blood pressure	1980.235	0.000
7	Number of hours since onset	978.578	0.000
3	Pain location	340.518	0.000
54	New ST segment depression	223.468	0.000
50	New Q wave	200.256	0.000
56	New T wave inversion	193.398	0.000
52	New ST segment elevation	188.224	0.000
55	Any ST segment depression	177.000	0.000
58	New intraventricular conduction defect	159.892	0.000
57	Any T wave inversion	151.672	0.000
39	Respiration rate	120.295	0.000
44	Diastolic murmur	117.165	0.000
59	Any intraventricular conduction defect	117.095	0.000

22	Prior angina prectoris	116.643	0.000
4	Chest pain radiation	114.724	0.000
38	Heart rate	109.556	0.000
51	Any Q wave	108.816	0.000
46	S3 gallop	105.087	0.000
18	Prior pain related to heart	101.630	0.000
19	Prior pain due to MI	89.169	0.000
47	S4 gallop	87.111	0.000
20	Prior pain due to angina prectoris	85.870	0.000
24	Congestive heart failure	85.266	0.000
43	Systolic murmur	84.497	0.000
26	Hiatal hernia	83.614	0.000
13	Dizziness/syncope	83.285	0.000
11	Palpitations	82.349	0.000
32	Beta blockers	79.733	0.000
37	Diastolic blood pressure	78.563	0.000
31	Nitrates	76.575	0.000
53	Any ST segment elevation	75.294	0.000
35	Antacids/H2 blockers	73.567	0.000
8	Duration of the last episode	63.415	0.000
33	Digitalis	60.113	0.000
41	Cyanosis	58.806	0.000
21	Prior MI	56.670	0.000
1	Age	56.543	0.000
40	Rales	54.042	0.000
48	Chest wall tenderness	53.812	0.000
5	Pain character	50.834	0.000
23	Prior atypical chest pain	49.971	0.000
9	Nausea	45.571	0.000
42	Pallor	45.247	0.000
14	Burping	43.215	0.000
27	Hypertension	41.360	0.000
16	Prior chest pain of this type	40.414	0.000
12	Dyspnea	36.725	0.000
34	Nonsteroidal anti-inflammatory	35.548	0.000
25	Peripheral vascular disease	28.190	0.000
49	Diaphoresis	27.678	0.000
6	Onset of pain	21.993	0.000
28	Diabetes	15.453	0.004
17	Physician consulted for prior pain	15.174	0.004
29	Smoker	15.063	0.005
45	Oedema	14.004	0.007
15	Palliative factors	12.859	0.012
2	Sex	10.844	0.028

4 Przedstawienie zastosowanego algorytmu

Zastosowanym przez nas algorytmem był algorytm k najbliższych sąsiadów (k-NN). Jest on zaliczany do grupy *lazy learning algorithms*, ponieważ nie potrzebuje on żadnych danych ze zbioru treningowego do wytworzenia modelu. Dane treningowe są w całości wykorzystywane w fazie testowania, co sprawia, że etap trenowania jest krótki, natomiast testowania - długi i kosztowny. Co więcej, algorytm ten jest nieparametryczny, co znaczy, że nigdy nie możemy pozbyć się danych treningowych, ponieważ parametry modelu zwiększają swoją liczebność z każdym nowym punktem zbioru treningowego.

4.1 Schemat

1. Załaduj dane treningowe oraz testowe.
2. Wybierz wartość dla k.
3. Wybierz metrykę odległości.
4. Dla każdego punktu p z danych testowych:
   (a) oblicz odległości punktu p do wszystkich punktów z danych treningowych korzystając z wybranej metryki,
   (b) zapisz wyliczone odległości do listy i posortuj ją rosnąco,
   (c) wybierz pierwsze k punktów z listy,
   (d) przyporządkuj punkt p do klasy, do której należy większość z obecnych k punktów.

4.2 Wykorzystane miary wyznaczania odległości

4.2.1 Odległość Manhattan

Nazwa tej metryki wywodzi się z najbardziej znanej dzielnicy Nowego Jorku, w której układ ulic poprzez swoje prostopadłe względem siebie ułożenie przypomina szachownicę. Wartość odległości dwóch punktów w tej metryce jest sumą wartości bezwzględnych różnic ich współrzędnych. Wzór przedstawiający tę zależność przedstawiony jest poniżej:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{2}$$

, gdzie n to liczba wymiarów przestrzeni, x oraz y są punktami, a i to analizowany wymiar.

4.2.2 Odległość Euklidesowa

Odległość Euklidesowa między dwoma punktami jest długością odcinka łączącego oba te punkty. Wartość odległości jest równa pierwiastkowi kwadratowemu sumy różnic pomiędzy wartościami współrzędnych podniesionymi do kwadratu we wszystkich wymiarach. Odległość w tej metryce wyrażona jest poniższym wzorem:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3}$$

, gdzie n to liczba wymiarów przestrzeni, x oraz y są punktami, a i to analizowany wymiar.

5 Opis środowiska programistycznego

Implementacja została przeprowadzona w całości z wykorzystaniem **języka Python** oraz bibliotek takich jak *numpy*[4], *pandas*[5] oraz *scikit-learn*[6]. Biblioteka *numpy* pozwoliła nam złączyć załadowane pliki w jedną listę, służyła do przechowywania danych oraz operacji na nich. *Pandas* - służyła głównie do ładnej prezentacji danych, a *scikit-learn* (biblioteka do uczenia maszynowego) umożliwiła wyznaczenie rankingu cech pod względem ich przydatności do klasyfikacji oraz pozwoliła na zaimplementowanie algorytmu k-NN. Z biblioteki *scikit-learn* wykorzystywane są metody użytkowe do oceny działania stworzonego modelu klasyfikacji. Do stworzenia implementacji zostało wykorzystane zintegrowane środowisko programistyczne **Pycharm**.

6 Przedstawienie testów statystycznych

W celu porównania wydajności dwóch modeli został zastosowany test t-studenta dla danych pochodzących z 5 razy powtarzanej metody 2-krotnej walidacji krzyżowej. Parowy test t-studenta dla 5x2 walidacji krzyżowej został zaproponowany przez Dietterichа [7], w celu wyeliminowania niedociągnięć w innych metodach, takich jak test t-studenta ponownie próbkowanego sparowanego testu. W sparowanym teście t-studenta powtarzamy podział za każdym razem tak, aby mieć połowę danych treningowych, a drugą połowę danych testowych, wziętych proporcjonalnie z całego zbioru danych. W naszym programie wielokrotny walidator krzyżowy *RepeatedStratifiedKFold(n_repeats=5, n_splits=2)* z biblioteki *scikit-learn* zapewnia zachowanie procentowego rozkładu klas pomiędzy podzbiorami.

Poniżej został przedstawiony wzór opisujący test t-studenta dla 5-krotnie powtarzanej metody 2-krotnej walidacji krzyżowej.[7]

$$t = \frac{p_1}{\sum_{i=1}^{5}\sqrt{(1/5)s_i^2}} \tag{4}$$

, gdzie p_1 to różnica z pierwszej iteracji z pierwszego podziału, s_i^2 to wariancja obliczana dla każdej iteracji (do 5-tej).

```
def paired_5x2_ttest(m1, m2):
    p_1_1 = m1['scores'][0] - m2['scores'][0]
    variances = []
    for i in range(0, 5 * 2, 2):  # 5 repeats of 2-fold CV
        p1 = m1['scores'][i] - m2['scores'][i]
        p2 = m1['scores'][i + 1] - m2['scores'][i + 1]

        variance = statistics.variance([p1, p2])
        variances.append(variance)

    t = p_1_1 / (math.sqrt(1 / 5.0 * sum(variances)))
    pvalue = stats.t.sf(np.abs(t), 5) * 2.0
    return t, pvalue
```

Listing 10: Funkcja przeprowadzająca parowy test t-studenta dla danych pochodzących z 5-krotnej 2-walidacji krzyżowej

Funkcja ta wyznacza wartość statystyki t-studenta, według wzoru 4. Założeniem działania tej funkcji jest wykonane na zbiorze danych powtórzona 5 razy dwukrotna walidacja krzyżowa. Dzięki wykorzystaniu tego typu walidacji można uzyskać założenie o niezależności prób, które jest wymagane do wykonania tego testu statystycznego. W teście statystycznym porównywane są średnie wartości dokładności stworzonych modeli algorytmu k-NN. W implementacji do przechowywania pośrednich wyników działania modelu została wykorzystana lista, która zawiera spłaszczoną strukturę 10-ciu wyników dokładności modelu k-NN. Wraz z wynikiem statystyki t-studenta, metoda zwraca p-value. Jest ona następnie zestawiana z wybranym dla eksperymentów poziomem istotności statystycznej $\alpha = 0.05$, co pozwala na zdecydowanie o przyjęciu lub odrzuceniu hipotezy zerowej o równości średnich dokładności działania algorytmu k-NN. W przypadku, gdy $p - value > \alpha$ nie możemy odrzucić hipotezy zerowej, a zatem stwierdzić, że różnica wyniku działania modeli jest statystycznie istotna [8]. Gdy $p - value < \alpha$ możemy odrzucić hipotezę zerową, czyli uznać różnicę działania modeli za istotną statystycznie.

```
def compare_two_models(model_idx1, model_idx2, df)
```

Listing 11: Funkcja porównująca dwa modele o podanych indeksach

Funkcja o sygnaturze z listingu 11 służy do sprawdzenia czy średnie wyniki dokładności działania dwóch wybranych modeli k-NN są różne na mocy zastosowania parowego testu statystycznego t-Studenta z założeniami przedstawionymi przez Diettricha.

```
def find_best_statistically_significant_model(df)
```

Listing 12: Funkcja znajdująca pierwszą różniącą się statystycznie parę modeli, zaczynając od modelu o najlepszej dokładności

Funkcja o sygnaturze z listingu 12 znajduje pierwszą parę modeli, których różnica średnich dokładności jest istotna statystycznie, Modele sprawdzane są od najlepszego co do dokładności - implementacja sortuje wewnątrz funkcji zadaną ramkę danych według kolumny z wartościami średnimi dokładności modelu.

```
def compare_every_model_paired(df)
```

Listing 13: Funkcja porównująca wszystkie pary dwóch modeli

Funkcja o sygnaturze z listingu 13 tworzy zestawienie porównawcze wszystkich modeli k-NN co do różnicy średnich dokładności działania modelu. Zestawienie to zawiera również informacje o liczbie par z różnicą w średnich istotną statystycznie oraz takich, dla których nie można było odrzucić hipotezy zerowej (nieistotnych statystycznie).

7 Przedstawienie planu eksperymentu oraz jego wyników

7.1 Podstawowe założenia

- ewaluacja wykorzystanego klasyfikatora przy pomocy 5 razy powtarzanej metody 2-krotnej walidacji krzyżowej
- mierzenie jakości klasyfikacji poprzez częstość poprawnych rozpoznań na zbiorze testującym
- wykorzystanie 3 różnych wartości k: 1, 5 oraz 10
- wykorzystanie 2 różnych miar odległości: manhattan oraz euklidesowej
- badania należy przeprowadzić dla różnej liczby cech - rozpoczynając od jednej - najlepszej wg. wyznaczonego rankingu, a następnie dokładać kolejno po jednej
- przeprowadzenie analizy statystycznej uzyskanych wyników

7.2 Plan eksperymentu

Na początku utworzone zostały stałe odpowiadające założeniom z sekcji 7.1. Jako walidator wykorzystaliśmy *RepeatedStratifiedKFold* z biblioteki *scikit-learn*, który jako parametry przyjął liczbę powtórzeń (*n_repeats*), liczbę zbiorów, na które dzielono zbiór główny (*n_splits*) oraz zmienną pozwalającą odtworzyć zachodzącą losowość wyników (*random_state*). Metoda ta powtarza wywołanie walidatora *StrarifieldKFold* wybraną ilość razy przy różnej losowości.

```
    rskf = RepeatedStratifiedKFold(n_repeats=5, n_splits=2, random_state=1)
    n_neighbors_variants = [1, 5, 10]
    metric_variants = ['manhattan', 'euclidean']
    number_of_features = ordered_features.shape[1] + 1  # or set to 8
```

Listing 14: Stworzenie stałych na podstawie założeń

Następnie przygotowaliśmy obiekt typu *DataFrame* w celu przechowywania wyników eksperymentu. Zapisywaliśmy do niego liczbę cech, którą braliśmy pod uwagę w eksperymencie, liczbę sąsiadów oraz metrykę odległości, częstość poprawnych rozwiązań dla każdego przypadku, uśrednioną częstość poprawnych rozwiązań oraz średnią macierz konfuzji - wygenerowaną poprzez uśrednienie wyników wszystkich macierzy wygenerowanych dla konkretnego badania.

```
    df_columns = ['n_features', 'n_neighbors', 'metric', 'scores', '
    ↪ mean_accuracy', 'mean_confusion_matrix']
    results_df = pd.DataFrame(columns=df_columns)
```

Listing 15: Przygotowanie struktury do przechowywania wyników

Algorytm składa się z 4 zagnieżdżonych iteracji, które ustawiają kolejno liczbę cech, liczbę sąsiadów, stosowaną metrykę oraz dokonują ewaluacji wykorzystanego klasyfikatora przy pomocy 5 razy powtarzanej metody 2-krotnej walidacji krzyżowej. Dla każdego zestawu składającego się z określonej liczby cech, liczby sąsiadów oraz metryki zapisywane są uśrednione wartości częstości poprawnych rozwiązań oraz uśredniona macierz konfuzji obliczone z powstałych w wyniku walidacji krzyżowej wartości.

```
for n_features in range(1, number_of_features):
    for n_neighbors in n_neighbors_variants:
        for metric in metric_variants:
            knn = KNeighborsClassifier(n_neighbors=n_neighbors, metric=metric)
            current_iteration_scores = []
            current_iteration_confusion_matrices = np.zeros(shape=(5, 5))
            number_of_iterations = 0

            for train, test in rskf.split(ordered_features[:, 0:n_features], classes):
                knn.fit(ordered_features[:, 0:n_features][train], classes[train])
                current_score = knn.score(ordered_features[:, 0:n_features][test], classes[test])
                current_iteration_scores.append(current_score)

                y_pred = knn.predict(ordered_features[:, 0:n_features][test])
                current_confusion_matrix = confusion_matrix(classes[test], y_pred=y_pred)
                current_iteration_confusion_matrices += current_confusion_matrix

                number_of_iterations += 1

            results_df.loc[len(results_df)] = [n_features, n_neighbors, metric, current_iteration_scores,
                                               np.array(current_iteration_scores).mean().round(3),
                                               (current_iteration_confusion_matrices / number_of_iterations)]
```

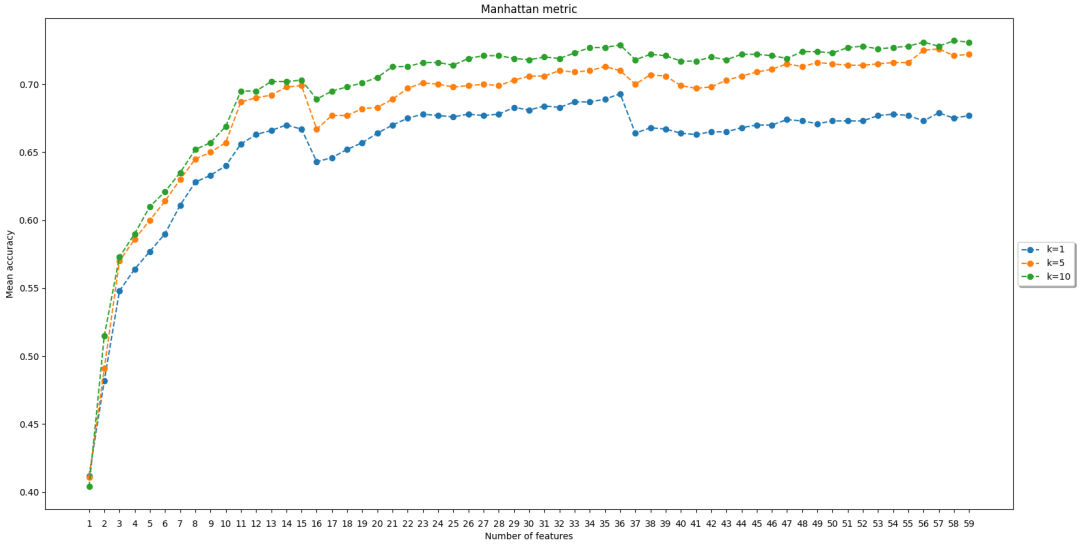
Listing 16: Algorytm eksperymentowania

Test t-studenta dla danych pochodzących z 5 razy powtarzanej metody 2-krotnej walidacji krzyżowej został opisany w sekcji 6.

7.3 Otrzymane wyniki

Tablica 4: Tabela częstości prawidłowych rozpoznań w zależności od liczby cech dla metryki odległości *Manhattan*

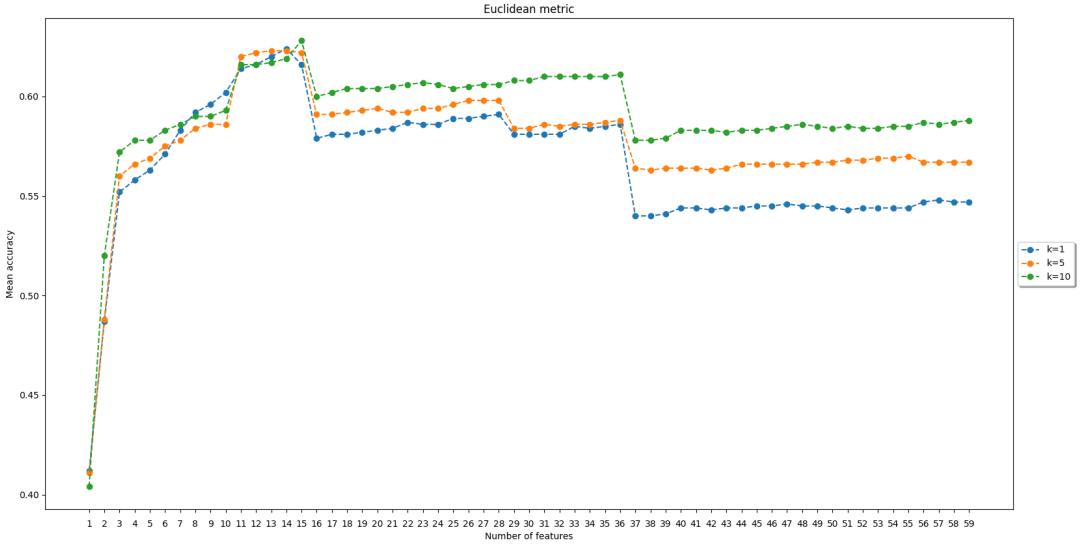
Liczba cech	Dokładność dla k=1	Dokładność dla k=5	Dokładność dla k=10
1	0.412	0.411	0.404
2	0.482	0.491	0.515
3	0.548	0.570	0.573
4	0.564	0.586	0.590
5	0.577	0.600	0.610
6	0.590	0.614	0.621
7	0.611	0.630	0.635
8	0.628	0.645	0.652
9	0.633	0.650	0.657
10	0.640	0.657	0.669
11	0.656	0.687	0.695
12	0.663	0.690	0.695
13	0.666	0.692	0.702
14	0.670	0.698	0.702
15	0.667	0.699	0.703
16	0.643	0.667	0.689
17	0.646	0.677	0.695
18	0.652	0.677	0.698
19	0.657	0.682	0.701
20	0.664	0.683	0.705
21	0.670	0.689	0.713
22	0.675	0.697	0.713
23	0.678	0.701	0.716
24	0.677	0.700	0.716
25	0.676	0.698	0.714
26	0.678	0.699	0.719
27	0.677	0.700	0.721
28	0.678	0.699	0.721
29	0.683	0.703	0.719
30	0.681	0.706	0.718
31	0.684	0.706	0.720
32	0.683	0.710	0.719
33	0.687	0.709	0.723
34	0.687	0.710	0.727
35	0.689	0.713	0.727
36	0.693	0.710	0.729
37	0.664	0.700	0.718
38	0.668	0.707	0.722
39	0.667	0.706	0.721
40	0.664	0.699	0.717
41	0.663	0.697	0.717
42	0.665	0.698	0.720
43	0.665	0.703	0.718
44	0.668	0.706	0.722
45	0.670	0.709	0.722
46	0.670	0.711	0.721
47	0.674	0.715	0.719
48	0.673	0.713	0.724
49	0.671	0.716	0.724
50	0.673	0.715	0.723
51	0.673	0.714	0.727
52	0.673	0.714	0.728
53	0.677	0.715	0.726
54	0.678	0.716	0.727
55	0.677	0.716	0.728
56	0.673	0.725	0.731
57	0.679	0.726	0.728
58	0.675	0.721	0.732
59	0.677	0.722	0.731



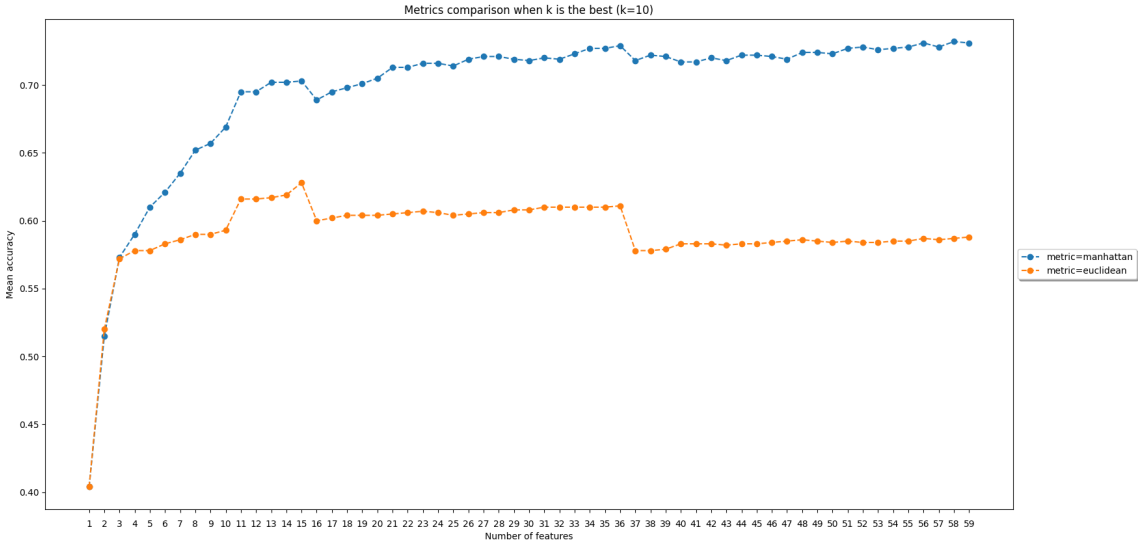
Rysunek 2: Wykres częstości poprawnych rozpoznań w zależności od liczby cech dla metryki odległości *Manhattan*

Tablica 5: Tabela częstości prawidłowych rozpoznań w zależności od liczby cech dla *euklidesowej* metryki odległości

Liczba cech	Dokładność dla k=1	Dokładność dla k=5	Dokładność dla k=10
1	0.412	0.411	0.404
2	0.487	0.488	0.520
3	0.552	0.560	0.572
4	0.558	0.566	0.578
5	0.563	0.569	0.578
6	0.571	0.575	0.583
7	0.583	0.578	0.586
8	0.592	0.584	0.590
9	0.596	0.586	0.590
10	0.602	0.586	0.593
11	0.614	0.620	0.616
12	0.616	0.622	0.616
13	0.620	0.623	0.617
14	0.624	0.623	0.619
15	0.616	0.622	0.628
16	0.579	0.591	0.600
17	0.581	0.591	0.602
18	0.581	0.592	0.604
19	0.582	0.593	0.604
20	0.583	0.594	0.604
21	0.584	0.592	0.605
22	0.587	0.592	0.606
23	0.586	0.594	0.607
24	0.586	0.594	0.606
25	0.589	0.596	0.604
26	0.589	0.598	0.605
27	0.590	0.598	0.606
28	0.591	0.598	0.606
29	0.581	0.584	0.608
30	0.581	0.584	0.608
31	0.581	0.586	0.610
32	0.581	0.585	0.610
33	0.585	0.586	0.610
34	0.584	0.586	0.610
35	0.585	0.587	0.610
36	0.586	0.588	0.611
37	0.540	0.564	0.578
38	0.540	0.563	0.578
39	0.541	0.564	0.579
40	0.544	0.564	0.583
41	0.544	0.564	0.583
42	0.543	0.563	0.583
43	0.544	0.564	0.582
44	0.544	0.566	0.583
45	0.545	0.566	0.583
46	0.545	0.566	0.584
47	0.546	0.566	0.585
48	0.545	0.566	0.586
49	0.545	0.567	0.585
50	0.544	0.567	0.584
51	0.543	0.568	0.585
52	0.544	0.568	0.584
53	0.544	0.569	0.584
54	0.544	0.569	0.585
55	0.544	0.570	0.585
56	0.547	0.567	0.587
57	0.548	0.567	0.586
58	0.547	0.567	0.587
59	0.547	0.567	0.588



Rysunek 3: Wykres częstości poprawnych rozpoznań w zależności od liczby cech dla *euklidesowej* metryki odległości



Rysunek 4: Wykres częstości poprawnych rozpoznań w zależności od liczby cech porównujący najlepszy dobór liczby sąsiadów dla metryk *Manhattan* oraz *euklidesowej*

Najlepsza średnia dokładność (najwyższa częstość poprawnych rozwiązań) otrzymana z algorytmu k-NN jest *0.732*. Została ona uzyskana dla modelu o parametrach:

- metryka - manhattan,

- liczba sąsiadów - 10,
- liczba cech - 58.

Rysunek 5: Uśredniona macierz konfuzji dla modelu o najlepszej dokładności

Na podstawie uśrednionej macierzy konfuzji (Rysunek 5) obliczono wartości precyzji, czułości oraz wynik f-1 korzystając z poniższych wzorów.

$$Precision = \frac{TruePositive}{ActualResults} = \frac{TruePositive}{TruePositive + FalsePositive} \tag{5}$$

$$Recall = \frac{TruePositive}{PredictedResults} = \frac{TruePositive}{TruePositive + FalseNegative} \tag{6}$$

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{7}$$

Tablica 6: Raport klasyfikacji najlepszych wyników

	precision	recall	f1-score
Angina prectoris	0.69	0.56	0.62
Angina prectoris - Prinzmetal variant	0.07	0.29	0.11
Myocardial infraction (subendocardial)	0.52	0.79	0.63
Myocardial infraction (transmural)	0.91	0.72	0.80
Pain of non-heart origin	0.93	0.87	0.90

Tablica 7: Pierwsza znaleziona para modeli, których różnica dokładności jest istotna statystycznie (dla danych posortowanych od modelu z najlepszą dokładnością)

k	m	f	accuracy	t-statistic	p-value	alfa
10	manhattan	58	0.732	12.925	$4.94 * 10^{-5}$	0.05
10	euclidean	59	0.588			

Oznaczenia w tabeli 7 przedstawiają następująco:

- k - liczbę sąsiadów,
- m - metrykę,
- f - liczbę cech,
- accuracy - dokładność,
- t-statistic - wartość statystyki testowej,
- p-value - prawdopodobieństwo testowe,
- alfa - poziomem istotności statystycznej.

```
Comparing [k=10, m=manhattan, f=56] with [k=5, m=euclidean, f=59]
    t-statistic: 9.042676616578628, p-value: 0.00027639692211995053, alfa: 0.05 -> 0.00027639692211995053 (p) < 0.05 (a)
Comparing [k=10, m=manhattan, f=56] with [k=5, m=euclidean, f=58]
    t-statistic: 9.681592813752752, p-value: 0.00019960250892234023, alfa: 0.05 -> 0.00019960250892234023 (p) < 0.05 (a)
Comparing [k=10, m=manhattan, f=56] with [k=5, m=euclidean, f=57]
    t-statistic: 10.328636594226598, p-value: 0.000146363070400347, alfa: 0.05 -> 0.000146363070400347 (p) < 0.05 (a)
Comparing [k=10, m=manhattan, f=56] with [k=10, m=euclidean, f=57]
    t-statistic: 13.026233358685406, p-value: 4.755220906043926e-05, alfa: 0.05 -> 4.755220906043926e-05 (p) < 0.05 (a)
Comparing [k=10, m=manhattan, f=56] with [k=10, m=euclidean, f=58]
    t-statistic: 13.844204144251238, p-value: 3.531729437665856e-05, alfa: 0.05 -> 3.531729437665856e-05 (p) < 0.05 (a)
Comparing [k=10, m=manhattan, f=56] with [k=10, m=euclidean, f=56]
    t-statistic: 13.211914926840217, p-value: 4.437931328049361e-05, alfa: 0.05 -> 4.437931328049361e-05 (p) < 0.05 (a)
Comparing [k=10, m=manhattan, f=56] with [k=10, m=euclidean, f=59]
    t-statistic: 16.66098224442123, p-value: 1.4229267016802309e-05, alfa: 0.05 -> 1.4229267016802309e-05 (p) < 0.05 (a)
Comparing [k=10, m=manhattan, f=58] with [k=1, m=euclidean, f=59]
    t-statistic: 5.513632021687024, p-value: 0.00268573610355514, alfa: 0.05 -> 0.00268573610355514 (p) < 0.05 (a)
Comparing [k=10, m=manhattan, f=58] with [k=5, m=euclidean, f=59]
    t-statistic: 8.005671062883629, p-value: 0.0004912715962336415, alfa: 0.05 -> 0.0004912715962336415 (p) < 0.05 (a)
Comparing [k=10, m=manhattan, f=58] with [k=10, m=euclidean, f=58]
    t-statistic: 11.175983431346818, p-value: 0.00010007471852489105, alfa: 0.05 -> 0.00010007471852489105 (p) < 0.05 (a)
Comparing [k=10, m=manhattan, f=58] with [k=10, m=euclidean, f=59]
    t-statistic: 12.9247627608975, p-value: 4.940055969794068e-05, alfa: 0.05 -> 4.940055969794068e-05 (p) < 0.05 (a)
Statistical significant pairs = 34271, statistical insignificant pairs = 28210
```

Rysunek 6: Zestawienie porównawcze wszystkich par modeli - zrzut konsoli z wynikami

Tablica 8: Pary modeli istotne i nieistotne statystycznie

Statistical significant pairs	Statistical insignificant pairs
34271	28210

Tablica 9: Porównanie modelu samego ze sobą

k	m	f	accuracy	t-statistic	p-value	alfa
10	manhattan	56	0.731	0	1	0.05
10	manhattan	56	0.731			

Tablica 10: Porównanie pary modeli, których różnica dokładności jest istotna statystycznie

k	m	f	accuracy	t-statistic	p-value	alfa
10	manhattan	54	0.727	3.244	0.0229	0.05
10	manhattan	22	0.713			

8 Omówienie otrzymanych wyników

Rysunki 2 oraz 3 przedstawiają jak zmieniała się częstość poprawnych rozpoznań w zależności od liczby cech oraz wartości parametru k algorytmu *k-NN* dla poszczególnych metryk.

Zarówno dla metryki *euklidesowej*, jak i *Manhattan* największa dokładność występowała przy wartości parametru k (liczbie sąsiadów) równej 10. Co więcej, w obu przypadkach najmniejsza dokładność występowała dla k równego 1, a wyniki dla k równego 5 znajdowały się

pomiędzy wynikami dla wspomnianych już wartości. Oba te rysunki pozwalają wysnuć wniosek, że dla wykorzystywanego zbioru danych większa wartość k oznaczała większą częstość poprawnych rozwiązań algorytmu.

Na wykresie przedstawionym na Rysunku 3 można zauważyć, że dla metryki *euklidesowej*, bez względu na wartość k, dokładność rozpoznań rosła do pewnej ilości cech (14), po czym zaczęła stopniowo i sukcesywnie maleć. Ta zależność jednak nie jest prawdziwa dla metryki *Manhattan* (Rysunek 2). Można zaobserwować, że wybrana metryka bardzo dobrze radzi sobie z większą liczbą wymiarów, gdyż najlepszy uzyskany przy jej pomocy wynik wystąpił dla 58 cech.

Na Rysunku 4 przedstawione zostało porównanie pomiędzy wynikami uzyskanymi przy wykorzystaniu obu metryk, które jasno pokazuje, że wybór metryki *Manhattan* powoduje dużo lepsze wyniki. Dzieje się tak, ponieważ w przestrzeniach o dużej liczbie wymiarów stosunek odległości do najbliższego punktu oraz najdalszego punktu jest bliski 1, co oznacza, że są one niemalże równe. Jest to szczególnie widoczne dla metryki *euklidesowej*, która wykorzystuje rzeczywistą odległość między dwoma punktami. Założenie, że punkty położone bliżej lepiej opisują rozpatrywany punkt niż punkty położone dalej traci wówczas sens. Badanie przedstawione w artykule *On the Suprising Behavior of Distance Metrics in High Dimensional Space*[1] ukazuje, że sensowność będącej uogólnioną miarą odległości między punktami odległości *Minkowskiego* przedstawionej wzorem:

$$L_m(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^m)^{1/m} \tag{8}$$

maleje szybciej wraz z rosnącą wymiarowością dla większych wartości m. Z tego też względu wykorzystanie metryki *Manhattan*, która przyjmuje 1 jako wartość parametru m powinno w badanym przypadku generować lepsze wyniki, niż przyjmująca 2 jako wartość parametru m metryka *euklidesowa*, co też jest obserwowane.

Wyniki dla najlepszego co do dokładności modelu, które są widoczne na Tablicy 6 oraz przedstawionej na Rysunku 5 macierzy konfuzji wskazują występującą zależność między liczebnością zbiorów (Sekcja 2.1), a częstością poprawnych rozpoznań danej choroby. Zauważyć można bowiem, iż najlepsze wyniki otrzymujemy dla najliczniejszych zbiorów, to jest *Pain of non-heart origin* oraz *Myocardial infraction (transmural)*, nieco gorsze dla tych o średniej liczebności - *Myocardial infraction (subendocardial)* i *Angina prectoris*, a zdecydowanie najgorsze, dla najmniej licznego zbioru - *Angina prectoris - Prinzmetal variant*. Dzieje się tak, ponieważ mamy tutaj do czynienia z danymi niezbalansowanymi, co przekłada się na zróżnicowaną ilość informacji o poszczególnych klasach w zbiorze treningowym.

Na Rysunku 6 została pokazana część zestawienia porównania wszystkich par modeli k-NN co do różnicy średnich dokładności działania modelu. Możemy zauważyć, że wartość statystyki t-studenta jest zawsze dodatnia. Wynika to z tego, że porównywane modele były posortowane malejąco według najwyższych dokładności. Porównanie dwóch modeli skutkowało zatem porównaniem modelu o wyższej dokładności z modelem o niższej dokładności. W rezultatach jest uwzględniona również informacja o wynikach, które są istotnie statystycznie oraz takich, które nie są istotne statystycznie, co obrazuje również tabela 8. Widzimy, że

istotnych statystycznie par modeli ($p - value < \alpha$) co do różnicy średnich dokładności działania dla danego modelu, jest więcej niż tych, które nie są istotne statystycznie (dla których nie można było odrzucić hipotezy zerowej).

W Tabeli 9 możemy zaobserwować porównanie modelu samego ze sobą - wartość t-statystyki wynosi 0, natomiast p-wartość jest równa 1. Posłużyło to sprawdzeniu działania stworzonej implementacji dla przypadków skrajnych.

W Tablicy 7 widzimy pierwszą znalezioną parę modeli, których różnica dokładności jest istotna statystycznie (dla danych posortowanych od modelu z najlepszą dokładnością). Możemy zauważyć, że $p - value < \alpha$, gdzie poziom istotności statystycznej $\alpha = 0.05$ przez co odrzucamy hipotezę zerową, czyli uznajemy różnicę działania modeli za istotną statystycznie. Wartość statystyki testowej jest natomiast większa od zera, ponieważ pierwszy model (metryka - manhattan, liczba sąsiadów - 10, liczba cech - 58, dokładność - 0.732) jest lepszy od drugiego (metryka - euclidean, liczba sąsiadów - 10, liczba cech - 59, dokładność - 0.588). Różnica tych modeli jest istotna statystycznie.
W tabeli 10 mamy przedstawione wyniki dla porównania dwóch modeli, których różnica dokładności jest również istotna statystycznie.

Dzięki malejącemu uporządkowaniu zebranych modeli od najlepszego co do średniej dokładności do najgorszego, przeprowadzenie testu statystycznego modelu o indeksie $n - 1$ (lepszy) z modelem o indeksie n (gorszy lub równy co do średniej dokładności z modelem o indeksie $n - 1$) pozwala stwierdzić, czy modele te faktycznie różnią się co do swojej dokładności. Zatem można stwierdzić, który model jest faktycznie lepszy. W przypadku, gdy różnica średnich dokładności modeli o indeksach $n - 1$ i n nie jest istotna statystycznie, nie możemy stwierdzić czy jeden z tych modeli jest faktycznie lepszy.

Najlepszym modelem, który różni się statystycznie z pewnym innym modelem oraz ma największą dokładność, jest więc model o parametrach: metryka - manhattan, liczba sąsiadów - 10, liczba cech - 58 oraz dokładności - 0.732. Zostało to stwierdzone na podstawie wyników średniej dokładności modelu oraz porównania tego modelu z innymi modelami poprzez wykorzystanie testów statystycznych. Model ten (o indeksie 0) nie różni się statystycznie z modelami o indeksach 1 - 224. Nie jesteśmy zatem w stanie stwierdzić, czy modele te różnią się, a zatem czy występuje różnica dokładności działania tych modeli w celu stwierdzenia czy dany model jest bezsprzecznie najlepszy.

Bibliografia

[1] Charu C. Aggarwal, Alexander Hinneburg i Daniel A. Keim. „On the Surprising Behavior of Distance Metrics in High Dimensional Space". W: *Database Theory — ICDT 2001*. Red. Jan Van den Bussche i Victor Vianu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, s. 420–434. ISBN: 978-3-540-44503-6.

[2] Wikipedia contributors. *Weryfikacja hipotez statystycznych*. https://pl.wikipedia.org/wiki/Weryfikacja_hipotez_statystycznych. Kw. 2019. (Term. wiz. 12. 11. 2020).

[3] Sampath K. Gajawada. *Chi-Square Test for Feature Selection in Machine learning*. https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223/. Paź. 2019. (Term. wiz. 12. 11. 2020).

[4] Charles R. Harris i in. „Array programming with NumPy". W: *Nature* 585.7825 (wrz. 2020), s. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[5] Wes McKinney. „Data Structures for Statistical Computing in Python". W: *Proceedings of the 9th Python in Science Conference*. Red. Stéfan van der Walt i Jarrod Millman. 2010, s. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[6] F. Pedregosa i in. „Scikit-learn: Machine Learning in Python". W: *Journal of Machine Learning Research* 12 (2011), s. 2825–2830.

[7] S. Raschka. *5x2cv paired t test*. http://rasbt.github.io/mlxtend/user_guide/evaluate/paired_ttest_5x2cv/. Kw. 2018. (Term. wiz. 12. 11. 2020).

[8] Pogotowie Statystyczne. *Istotność statystyczna. Poprawna interpretacja p value*. https://pogotowiestatystyczne.pl/istotnosc-statystyczna/. Lip. 2016. (Term. wiz. 12. 11. 2020).

[9] H. Wei. *Feature Selection Methods with Code Examples*. https://medium.com/analytics-vidhya/feature-selection-methods-with-code-examples-a78439477cd4/. Sierp. 2019. (Term. wiz. 12. 11. 2020).