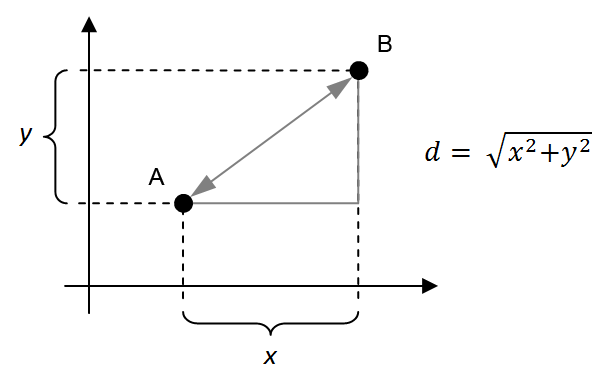
Metryka odległości:

W algorytmach odległościowych istotną rolę odgrywa wybrana metryka wg której mierzone będzie odległość pomiędzy dwoma badanymi punktami. Spośród wielu wybrane zostały:

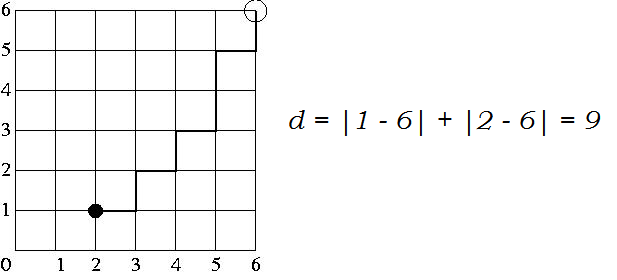
Metryka euklidesowa, w której za odległość między dwoma punktami w przestrzeni przyjmuje się pierwiastek euklidesowego iloczynu skalarnego różnicy dwóch wektorów:  


Przykład:

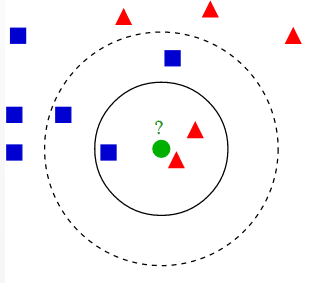


Metryka Manhattan, w której odległość dwóch punktów to suma wartości bezwzględnych różnic ich współrzędnych, zgodnie z poniższym wzorem:  


Przykład:



Algorytm *k*-NN pozwala wyszukać w zadanym zbiorze punktów, *k* punktów znajdujących się najbliżej nowego punktu, korzystając przy tym z wybranej miary odległości.   
W zagadnieniu klasyfikacji, algorytm *k-*NN przyjmuje jako argument tzw. zbiór uczący składający się z wielowymiarowych wektorów cech oraz przypisanych do nich klas. Faza klasyfikacji nowego obiektu polega na przypisaniu odpowiedniej klasy nieznanemu wektorowi cech. Nowy obiekt przypisywany jest do klasy, która występuje najczęściej wśród *k* najbliżej znajdujących się obiektów ze zbioru uczącego, zgodnie z wybraną metryką. Specjalnym przypadkiem algorytmu jest sytuacja, w której *k*= 1. Nazywa się algorytmem najbliższego sąsiada, w którym klasa nowego obiektu ustalana jest na podstawie najbliżej leżącej próbki ze zbioru danych uczących. Na poniższym rysunku przestawiona została przykładowa przestrzeń zawierająca dane uczące (niebieskie kwadraty oraz czerwona trójkąty) oraz niezidentyfikowaną próbkę – zielone kółko):



Przyjmując liczbę najbliższych sąsiadów *k = 3* (ciągła linia na rysunku), nowy obiekt zostanie przypisany do gruby czerwonych trójkątów, ponieważ jest ich więcej wśród 3 najbliższych próbek.  
Przyjmując *k = 5*, nowy obiekt zostanie przypisany do klasy niebieskich kwadratów, ponieważ w najbliższym otoczeniu znajdują się 3 kwadraty i tylko 2 trójkąty

*k*NN

**Implementacja**

Badania eksperymentalne z użyciem algorytmu *k* najbliższych sąsiadów wykonane zostały – zgodnie z wytycznymi – przy użyciu metody podwójnej walidacji krzyżowej, powtarzanej pięciokrotnie, uśredniając wyniki wszystkich eksperymentów. W tym celu zaimplementowany został algorytm przedstawiony na poniższym listingu:

Listing 1 Algorytm walidacji krzyżowej dla algorytmu kNN

Argumentami funkcji są w kolejności:

* *features* - Liczba cech branych pod uwagę, zgodnie z kolejnością wyznaczonego rankingu
* *neighbors* - Liczba najbliższych sąsiadów
* *standardize* - Logiczny parametr określający czy dane powinny być normalizowane
* *distanceMetric* – wybrana metryka pomiaru odległości

Ranking cech, którego sposób wyznaczenia opisany został w rozdziale **XXX,** przechowywany jest w wektorze *feature\_rank* . Pierwszym krokiem algorytmu jest załadowanie danych wejściowych wraz z przypisanymi do nich klasami, oraz wybranie z nich podanej w parametrze liczby najlepszych cech.  
Następnie w wykonującej się pięciokrotnie pętli, wyselekcjonowane dane dzielone są losowo na 2 równe podzbiory – uczący(*train*) oraz testujący(*test*), na podstawie których budowane są 2 klasyfikatory. Poniżej przedstawiona została implementacja funkcji wykorzystanej do budowy klasyfikatora:

Listing 2 Budowa klasyfikatora kNN

Następnie klasyfikator zbudowany na podstawie podzbioru *train* testowany jest na podzbiorze *test*, natomiast klasyfikator zbudowany na podstawie zbioru *test* testowany jest na podzbiorze *train.* Do celów testowania klasyfikatora wykorzystana została funkcja *predict* klasy *ClassificationKNN,* przyporządkowująca podanym danym testowym klasy, na podstawie podanego w parametrze wytrenowanego klasyfikatora. Metoda wykorzystana została w zaimplementowanej funkcji getScore, która jednocześnie zlicza poprawnie zaklasyfikowane obiekty:

Listing 3Testowanie oraz badanie jakości klasyfikatora kNN

**Opis badań**

Opisany w rozdziale **XXX** sposób implementacji algorytmu pozwolił na stworzenie skryptu, w którym algorytm *k*NN testowany jest z różnymi parametrami wywołania. Argumenty algorytmu w poniższym teście przyjmują wartości z następujących zakresów:

* *features*: <1, 31>
* *neighbors*: {1, 6, 13}
* *standardize*: {true, false}
* *distanceMetric*: {Euklidesowa, Manhattan}

Skrypt testujący sprawdza wszystkie możliwe kombinacje powyższych parametrów, generując wynikową macierz w postaci tabeli o wymiarach 31x12 komórek.

**Analiza wyników**

Liczba uwzględnianych cech.

Najniższa jakość klasyfikatora, niezależnie od innych parametrów, osiągana była przy uwzględnianiu najmniejszej liczby cech z rankingu. Wartość dla wszystkich 12 przypadków wywołań algorytmu z uwzględnieniem tylko pierwszej cechy z rankingu wahała się od: 12,68% (dla *neighbors* = 1, standerdize = true, distanceMetric = Manhattan) do 19,325 (dla *neighbors* = 13, standerdize = false, distanceMetric = Manhattan).

We wszystkich przypadkach najwyższy skok jakościowy został odnotowany po uwzględnieniu kolejnej cechy z rankingu. Dla dwóch najlepszych cech, wyniki wynosiły od 23,66% (dla *neighbors* = 1, standerdize = false, distanceMetric = Euklidesowa), do 34,44% (dla *neighbors* = 13, standerdize = false, distanceMetric = Euklidesowa).

Poniższy wykres przedstawia wykres zależności ilości branych pod uwagę cech od skuteczności klasyfikacji, dla czterech opisanych wyżej przypadków.

**Weryfikacja algorytmu**

W celu weryfikacji poprawności działania zaimplementowanego algorytmu oraz skryptu testującego, przeprowadzone zostało analogiczne badanie, w którym klasyfikatory testowane były tymi samymi danymi, na których były uczone. Zgodnie z przewidywaniami, jakoś wyników znacznie wzrosła, utrzymując się na poziomie 90% - 100% poprawnych wyników dla wszystkich możliwych kombinacji parametrów. Test ten potwierdza poprawność działania samych klasyfikatorów oraz procesu ich uczenia.