KNN CLASSIFIER

Generated by Doxygen 1.9.3

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 functions.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <algorithm>
#include <sstream>
#include <map>
#include <vector>
#include "functions.h"
```

Functions

- int argInfo (int &argc, char ∗argv[ ], string &addressTrain, string &addressTest, string &addressOut, int &k)
- int argValidate (const string &addressTest, const string &addressTrain, const string &addressOut)
- int additionalValid (int k, int trainingAmount)
- void readData (const string &addressTrain, const string &addressTest, int &dimensions, int &trainingAmount, int &testAmount, double ∗∗&trainingPoints, double ∗∗&testPoints, string ∗&trainingPointClasses)
- void checkDistances (double ∗∗&testPoints, double ∗∗&trainingPoints, double ∗∗&distances, int testAmount, int trainAmount, int dimensions)
- void findTestClasses (map< string, int > &classes, double ∗∗&distances, string ∗&testPointClasses, string ∗&trainingPointClasses, int testAmount, int k, int trainAmount)
- void writeToFile (const string &address, string ∗&testPointsClasses, double ∗∗&testPoints, int testAmount, int dimensions)
- void deleteTables (string ∗&testPointClasses, string ∗&trainingPointClasses, double ∗∗&testPoints, double ∗∗&trainingPoints, double ∗∗&distances)

2.1.1 Function Documentation

Zawiera instrukcje wszystkich funkcji programu.

2.1.1.1 additionalValid()

```
int additionalValid (
            int k,
            int trainingAmount )
```

Dodatkowa walidacja argumentu k, który nie może być większy od ilości punktów treningowych

Parameters

k	ilość k najbliższych sąsiadów
trainingAmount	ilość punktów treningowych

Returns

zwraca błąd i kończy program jeśli k jest większe od ilości punktów treningowych

2.1.1.2 argInfo()

```
int argInfo (
            int & argc,
            char * argv[],
            string & addressTrain,
            string & addressTest,
            string & addressOut,
            int & k )
```

Funkcja przyporządkowywuje wartości a tablicy argv odpowiednim zmiennym które będą używane w całym programie

Parameters

argc	ilość argumentów podanych przez użytkownika
argv	tablica z argumentami podanymi przez użytkownika
addressTrain	adres pliku z punktami treningowymi podany przez użytkownika
addressTest	adres pliku z punktami testowymi podany przez użytkownika
addressOut	adres pliku wyjściowego wybranego przez użytkownika
k	ilość najbliższych sąsiadów służąca do dalszej klasyfikacji

Returns

jeśli ilość argumentów jest odpowiednia to po wykonaniu funkcji zwrócone zostaną wypełnione zmienne, jeśli nie to funkcja zwróci błąd

2.1.1.3 argValidate()

```
int argValidate (
            const string & addressTest,
            const string & addressTrain,
            const string & addressOut )
```

Funkcja odpowiada za prostą walidacje argumentów podanych przez użytkownika

Sprawdza najczęstsze błędy związane z niepodaniem rozszerzenia pliku

Parameters

addressTest	adres pliku z punktami testowymi
addressTrain	adres pliku z punktami treningowymi
addressOut	adres pliku wyjściowego

Returns

funkcja kończy się bez błędu jeśli wszystkie pliki mają rozszerzenie txt, jeśli pojawia się błąd - program przerywa się

2.1.1.4 checkDistances()

```
void checkDistances (
            double **& testPoints,
            double **& trainingPoints,
            double **& distances,
            int testAmount,
            int trainAmount,
            int dimensions )
```

Funkcja sprawdza odległość każdego punktu testowego do każdego z punktów treningowych za pomocą metody euklidesowej

Odległości zapisywane są do tablicy distances która ma wielkość ilość punktów testowych na ilość punktów treningowych

Parameters

testPoints	tablica zawierająca współrzędne punktów testowych
trainingPoints	tablica zawierające współrzędne punktów treningowych
distances	tablica zawierająca odległości kolejnych punktów testowych od każdego punktu treningowego
testAmount	ilość punktów testowych
trainAmount	ilość punktów treningowych
dimensions	ilość wymiarów D

2.1.1.5 deleteTables()

```
void deleteTables (
            string *& testPointClasses,
            string *& trainingPointClasses,
            double **& testPoints,
            double **& trainingPoints,
            double **& distances )
```

Funkcja usuwa wszystkie tablice zaalokowane wcześniej dynamicznie

Parameters

testPointClassess	tablica klas punktów testowych
trainingPointClassess	tablica klas punktów treningowych
testPoints	talica współrzędnych punktów testowych
trainingPoints	tablica współrzędnych punktów treningowych
distances	tablica dystansów kolejnych punktów testowych od wszystkich punktów treningowych

2.1.1.6 findTestClasses()

```
void findTestClasses (
            map< string, int > & classes,
            double **& distances,
            string *& testPointClasses,
            string *& trainingPointClasses,
            int testAmount,
            int k,
            int trainAmount )
```

Funkcja wypełnia tablicę klas punktów testowych na podstawie k najbliższych puntków treningowych względem kolejnych punktów testowych, klasy zapisywane są do mapy classes

Na podstawie ilości wystąpień wybranych klas, ta która ma największą wartość w mapie przyporządkowywana jest do wybranego punktu testowego

Parameters

classes	mapa do której zapisywane będą klasy k najbliższych punktów treningowych do kolejnych punktów testowych
distances	tablica odległości kolejnych punktów testowych od każdego punktu treningowego
testPointClasses	tablica klas punktów testowych, którą wypełnia ta funkcja
trainingPointClasses	tablica klas punktów treningowych
testAmount	ilość punktów testowych
k	liczba k najbliższych sąsiadów (w tym przypadku najbliższych, względem kolejnych punktów testowych, punktów treningowych)
trainAmount	liczba punktów treningowych

2.1.1.7 readData()

```
void readData (
            const string & addressTrain,
            const string & addressTest,
            int & dimensions,
            int & trainingAmount,
            int & testAmount,
```

```
            double **& trainingPoints,
            double **& testPoints,
            string *& trainingPointClasses )
```

Funkcja zczytuje współrzędne punktów treningowych i testowych z odpowiednich plików

Po zczytaniu danych zapisywane są one do odpowiednich tablic

Parameters

addressTrain	adres pliku z punktami treningowymi
addressTest	adres pliku z punktami testowymi
dimensions	liczba wymiarów D w których zapisane są wszystkie punkty
trainingAmount	ilość punktów treningowych
testAmount	ilość punktów testowych
trainingPoints	tablica D wymiarowa zawierająca współrzędne wszystkich punktów treningowych (trainingAmount x dimensions)
testPoints	tablica D wymiarowa zawierająca współrzędne wszystkich punktów testowtch (testAmount x dimensions)
trainingPointClasses	tablica jednowymiarowa stringów zawierająca klasy punktów treningowych

2.1.1.8 writeToFile()

```
void writeToFile (
            const string & address,
            string *& testPointsClasses,
            double **& testPoints,
            int testAmount,
            int dimensions )
```

Funkcja wypisuje informacje do pliku w sposób podobny do pliku z punktami testowymi, lecz w tym wypadku każdemu z punktów przyporządkwywuje ona klase dobraną w wyniku funkcji findTestClasses()

Parameters

address	adres pliku do którego zostaną wypisane informacje
testPointsClasses	tablica z klasami punktów testowych
testPoints	tablica z współrzędnymi punktów testowych
testAmount	ilość punktów testowych
dimensions	ilość wymiarów D

2.2 functions.h File Reference

```
#include <iostream>
#include <fstream>
#include <algorithm>
```

```
#include <sstream>
#include <map>
```

Functions

- int argInfo (int &argc, char ∗argv[ ], string &addressTrain, string &addressTest, string &addressOut, int &k)
- int argValidate (const string &addressTest, const string &addressTrain, const string &addressOut)
- int additionalValid (int k, int trainingAmount)
- void readData (const string &addressTrain, const string &addressTest, int &dimensions, int &trainingAmount, int &testAmount, double ∗∗&trainingPoints, double ∗∗&testPoints, string ∗&trainingPointClasses)
- void checkDistances (double ∗∗&testPoints, double ∗∗&trainingPoints, double ∗∗&distances, int testAmount, int trainAmount, int dimensions)
- void findTestClasses (map< string, int > &classes, double ∗∗&distances, string ∗&testPointClasses, string ∗&trainingPointClasses, int testAmount, int k, int trainAmount)
- void writeToFile (const string &address, string ∗&testPointsClasses, double ∗∗&testPoints, int testAmount, int dimensions)
- void deleteTables (string ∗&testPointClasses, string ∗&trainingPointClasses, double ∗∗&testPoints, double ∗∗&trainingPoints, double ∗∗&distances)

2.2.1 Function Documentation

Zawiera nagłówki wszystkich funkcji programu.

2.2.1.1 additionalValid()

```
int additionalValid (
            int k,
            int trainingAmount )
```

Dodatkowa walidacja argumentu k, który nie może być większy od ilości punktów treningowych

Parameters

k	ilość k najbliższych sąsiadów
trainingAmount	ilość punktów treningowych

Returns

zwraca błąd i kończy program jeśli k jest większe od ilości punktów treningowych

2.2.1.2 argInfo()

```
int argInfo (
            int & argc,
            char * argv[],
```

```
            string & addressTrain,
            string & addressTest,
            string & addressOut,
            int & k )
```

Funkcja przyporządkowywuje wartości a tablicy argv odpowiednim zmiennym które będą używane w całym programie

Parameters

argc	ilość argumentów podanych przez użytkownika
argv	tablica z argumentami podanymi przez użytkownika
addressTrain	adres pliku z punktami treningowymi podany przez użytkownika
addressTest	adres pliku z punktami testowymi podany przez użytkownika
addressOut	adres pliku wyjściowego wybranego przez użytkownika
k	ilość najbliższych sąsiadów służąca do dalszej klasyfikacji

Returns

jeśli ilość argumentów jest odpowiednia to po wykonaniu funkcji zwrócone zostaną wypełnione zmienne, jeśli nie to funkcja zwróci błąd

2.2.1.3 argValidate()

```
int argValidate (
            const string & addressTest,
            const string & addressTrain,
            const string & addressOut )
```

Funkcja odpowiada za prostą walidacje argumentów podanych przez użytkownika

Sprawdza najczęstsze błędy związane z niepodaniem rozszerzenia pliku

Parameters

addressTest	adres pliku z punktami testowymi
addressTrain	adres pliku z punktami treningowymi
addressOut	adres pliku wyjściowego

Returns

funkcja kończy się bez błędu jeśli wszystkie pliki mają rozszerzenie txt, jeśli pojawia się błąd - program przerywa się

2.2.1.4 checkDistances()

```
void checkDistances (
            double **& testPoints,
            double **& trainingPoints,
            double **& distances,
            int testAmount,
            int trainAmount,
            int dimensions )
```

Funkcja sprawdza odległość każdego punktu testowego do każdego z punktów treningowych za pomocą metody euklidesowej

Odległości zapisywane są do tablicy distances która ma wielkość ilość punktów testowych na ilość punktów treningowych

Parameters

testPoints	tablica zawierająca współrzędne punktów testowych
trainingPoints	tablica zawierające współrzędne punktów treningowych
distances	tablica zawierająca odległości kolejnych punktów testowych od każdego punktu treningowego
testAmount	ilość punktów testowych
trainAmount	ilość punktów treningowych
dimensions	ilość wymiarów D

2.2.1.5 deleteTables()

```
void deleteTables (
            string *& testPointClasses,
            string *& trainingPointClasses,
            double **& testPoints,
            double **& trainingPoints,
            double **& distances )
```

Funkcja usuwa wszystkie tablice zaalokowane wcześniej dynamicznie

Parameters

testPointClassess	tablica klas punktów testowych
trainingPointClassess	tablica klas punktów treningowych
testPoints	talica współrzędnych punktów testowych
trainingPoints	tablica współrzędnych punktów treningowych
distances	tablica dystansów kolejnych punktów testowych od wszystkich punktów treningowych

2.2.1.6 findTestClasses()

```
void findTestClasses (
```

```
            map< string, int > & classes,
            double **& distances,
            string *& testPointClasses,
            string *& trainingPointClasses,
            int testAmount,
            int k,
            int trainAmount )
```

Funkcja wypełnia tablicę klas punktów testowych na podstawie k najbliższych puntków treningowych względem kolejnych punktów testowych, klasy zapisywane są do mapy classes

Na podstawie ilości wystąpień wybranych klas, ta która ma największą wartość w mapie przyporządkowywana jest do wybranego punktu testowego

Parameters

classes	mapa do której zapisywane będą klasy k najbliższych punktów treningowych do kolejnych punktów testowych
distances	tablica odległości kolejnych punktów testowych od każdego punktu treningowego
testPointClasses	tablica klas punktów testowych, którą wypełnia ta funkcja
trainingPointClasses	tablica klas punktów treningowych
testAmount	ilość punktów testowych
k	liczba k najbliższych sąsiadów (w tym przypadku najbliższych, względem kolejnych punktów testowych, punktów treningowych)
trainAmount	liczba punktów treningowych

2.2.1.7 readData()

```
void readData (
            const string & addressTrain,
            const string & addressTest,
            int & dimensions,
            int & trainingAmount,
            int & testAmount,
            double **& trainingPoints,
            double **& testPoints,
            string *& trainingPointClasses )
```

Funkcja zczytuje współrzędne punktów treningowych i testowych z odpowiednich plików

Po zczytaniu danych zapisywane są one do odpowiednich tablic

Parameters

addressTrain	adres pliku z punktami treningowymi
addressTest	adres pliku z punktami testowymi
dimensions	liczba wymiarów D w których zapisane są wszystkie punkty
trainingAmount	ilość punktów treningowych
testAmount	ilość punktów testowych
trainingPoints	tablica D wymiarowa zawierająca współrzędne wszystkich punktów treningowych (trainingAmount x dimensions)
testPoints	tablica D wymiarowa zawierająca współrzędne wszystkich punktów testowtch (testAmount x dimensions)
trainingPointClasses	tablica jednowymiarowa stringów zawierająca klasy punktów treningowych

2.2.1.8 writeToFile()

```
void writeToFile (
            const string & address,
            string *& testPointsClasses,
            double **& testPoints,
            int testAmount,
            int dimensions )
```

Funkcja wypisuje informacje do pliku w sposób podobny do pliku z punktami testowymi, lecz w tym wypadku każdemu z punktów przyporządkwywuje ona klase dobraną w wyniku funkcji findTestClasses()

Parameters

address	adres pliku do którego zostaną wypisane informacje
testPointsClasses	tablica z klasami punktów testowych
testPoints	tablica z współrzędnymi punktów testowych
testAmount	ilość punktów testowych
dimensions	ilość wymiarów D

2.3 functions.h

Go to the documentation of this file.

```
1
2 //
3 // Created by Michin on 28.11.2021.
4 //
5 #ifndef PROJEKT_FUNCTIONS_H
6 #define PROJEKT_FUNCTIONS_H
7 #include <iostream>
8 #include <fstream>
9 #include <algorithm>
10 #include <sstream>
11 #include <map>
12 using namespace std;
23 int argInfo(int &argc, char * argv[], string &addressTrain, string &addressTest, string &addressOut, int
       &k);
32 int argValidate(const string &addressTest,const string &addressTrain, const string &addressOut);
39 int additionalValid(int k, int trainingAmount);
52 void readData(const string &addressTrain, const string &addressTest, int &dimensions, int
       &trainingAmount, int &testAmount, double ** &trainingPoints, double ** &testPoints, string *
       &trainingPointClasses);
63 void checkDistances(double ** &testPoints, double ** &trainingPoints, double ** &distances, int
       testAmount, int trainAmount, int dimensions);
75 void findTestClasses(map<string, int> &classes, double ** &distances, string * &testPointClasses, string
       * &trainingPointClasses, int testAmount, int k, int trainAmount);
84 void writeToFile(const string &address, string * &testPointsClasses, double ** &testPoints, int
       testAmount, int dimensions);
93 void deleteTables(string * &testPointClasses, string * &trainingPointClasses, double ** &testPoints,
       double ** &trainingPoints, double ** &distances);
94 #endif //PROJEKT_FUNCTIONS_H
```

2.4 main.cpp File Reference

```
#include <iostream>
#include <fstream>
```

```
#include <algorithm>
#include <sstream>
#include <map>
#include "functions.h"
```

Functions

- int main (int argc, char ∗argv[ ])

2.4.1 Function Documentation

Główny plik programu.

2.4.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Główna funkcja programu.

Parameters

argc	ilość argumentów podanych przez użytkownika
argv	tablica z argumentami

Index