

DiabetesAI

1.0.0

Создано системой Doxygen 1.9.7

1 Алфавитный указатель классов	1
1.1 Классы	1
2 Список файлов	2
2.1 Файлы	2
3 Классы	2
3.1 Класс DiabetesData	2
3.1.1 Подробное описание	3
3.1.2 Конструктор(ы)	3
3.1.3 Методы	4
3.1.4 Данные класса	5
3.2 Класс LogisticRegression	6
3.2.1 Подробное описание	7
3.2.2 Конструктор(ы)	7
3.2.3 Методы	7
3.2.4 Данные класса	11
3.3 Класс Plot	12
3.3.1 Подробное описание	13
3.3.2 Конструктор(ы)	13
3.3.3 Методы	13
4 Файлы	14
4.1 Файл prj.lab/diabetes_data/diabetes_data.cpp	14
4.2 Файл prj.lab/diabetes_data/include/diabetes_data/diabetes_data.hpp	15
4.2.1 Макросы	15
4.3 diabetes_data.hpp	15
4.4 Файл prj.lab/plot/include/plot/plot.hpp	16
4.4.1 Макросы	16
4.5 plot.hpp	16
4.6 Файл prj.lab/plot/plot.cpp	17
4.7 Файл prj.lab/regression/include/regression/regression.hpp	17
4.7.1 Макросы	17
4.8 regression.hpp	17
4.9 Файл prj.lab/regression/regression.cpp	18
4.10 Файл prj.test/main.cpp	18
4.10.1 Функции	18
Предметный указатель	19

1 Алфавитный указатель классов

1.1 Классы

Классы с их кратким описанием.

DiabetesData

Класс [DiabetesData](#) предназначен для подготовки датасета 2

LogisticRegression

Класс [LogisticRegression](#) предназначен для обучения модели на основе лог.регрессии 6

Plot

Класс [Plot](#) предназначен для создания .tex файла с информацией о модели 12

2 Список файлов

2.1 Файлы

Полный список файлов.

prj.lab/diabetes_data/ diabetes_data.cpp	14
prj.lab/diabetes_data/include/diabetes_data/ diabetes_data.hpp	15
prj.lab/plot/ plot.cpp	17
prj.lab/plot/include/plot/ plot.hpp	16
prj.lab/regression/ regression.cpp	18
prj.lab/regression/include/regression/ regression.hpp	17
prj.test/ main.cpp	18

3 Классы

3.1 Класс DiabetesData

Класс [DiabetesData](#) предназначен для подготовки датасета

```
#include <diabetes_data.hpp>
```

Открытые члены

- [DiabetesData](#) ()=default
- [DiabetesData](#) (const string &file_path, const string &data_name)
- [DiabetesData](#) (const vector< vector< double > > &features)
- void [load_data_from_file](#) (const string &file_path, const string &data_name)
- vector< vector< double > > [get_X](#) ()
- vector< int > [get_y](#) ()

Открытые статические члены

- static vector< vector< double > > [data_normalization](#) (vector< vector< double > > X)

Закрытые данные

- `vector< vector< double > > features_`
- `vector< vector< string > > dataset_`
- `vector< vector< double > > X_`
- `vector< int > y_`

3.1.1 Подробное описание

Класс `DiabetesData` предназначен для подготовки датасета

Класс `DiabetesData` предоставляет возможность подготовить датасет для дальнейшей работы с классом `LogisticRegression` и `Plot`, путем разбиения датасета на двумерный вектор `double` и одномерный вектор `int`: вектор `X_` содержит в себе строки датасета, не включая названия столбцов и исход(outcome), и `y_`, содержит в себе лишь исход(outcome)

3.1.2 Конструктор(ы)

`DiabetesData()` [1/3]

`DiabetesData::DiabetesData ()` [default]

Конструктор по умолчанию для создания объекта класса

`DiabetesData()` [2/3]

```
DiabetesData::DiabetesData (
    const string & file_path,
    const string & data_name ) [explicit]
```

Конструктор принимает на вход полный путь к директории с датасетами и название датасета, после чего он вызывает функцию `load_data_from_file`

Аргументы

in	<code>file_path</code>	Полный путь к репозиторию с датасетами вида: "C:\\...\\data"
in	<code>data_name</code>	Название файла .csv с датасетом вида: "dataset"

```
load_data_from_file(file_path, data_name);
```

`DiabetesData()` [3/3]

```
DiabetesData::DiabetesData (
    const vector< vector< double > > & features ) [explicit]
```

Конструктор принимает на вход выборку и нормализует её с помощью `data_normalization`

Аргументы

in	features	двумерный вектор double состоящий из одной строки и n столбцов (n = кол-ву столбцов датасета - 1)
----	----------	---

```
if (features.size() != dataset_[0].size() || features.empty())
    throw runtime_error(
        "The number of parameters does not match the number of dataset parameters, check the correctness of the entered data
        and check which dataset you submitted");
features_ = data_normalization(features);
```

3.1.3 Методы

data_normalization()

```
vector< vector< double > > DiabetesData::data_normalization (
    vector< vector< double > > X ) [static]
```

Функция принимает на вход выборку $X_{_}$ и нормализует её методом z-масштабирования

Аргументы

X	Двумерный вектор типа double, содержащий в себе выборку
---	---

```
vector<double> avarage;
for (int j = 0; j < X[0].size(); j++) {
    double summ = 0;
    for (auto &i: X) {
        summ += i[j];
    }
    avarage.push_back(summ / X.size());
}

vector<double> deviation;
for (int j = 0; j < X[0].size(); j++) {
    double summ = 0;
    for (auto &i: X) {
        summ += pow((i[j] - avarage[j]), 2);
    }
    deviation.push_back(sqrt(summ / (X.size() - 1)));
}

for (int j = 0; j < X[0].size(); j++) {
    for (auto &i: X) {
        i[j] = (i[j] - avarage[j]) / deviation[j];
        if (i[j] > 3 * deviation[j]) {
            i[j] = 0;
        }
    }
}
return X;
```

get_X()

```
vector< vector< double > > DiabetesData::get_X ( )
```

Функция возвращающая приватный член $X_{_}$

Возвращает

двумерный вектор типа double

```
return X_;
```

get_y()

```
vector< int > DiabetesData::get_y ( )
```

Функция возвращающая приватный член y_

Возвращает

вектор типа int

```
return y_;
```

load_data_from_file()

```
void DiabetesData::load_data_from_file (
    const string & file_path,
    const string & data_name )
```

Функция принимает на вход полный путь к директории с датасетами и название датасета, обрабатывает его, и записывает результат в X_ и y_

Аргументы

file_path	Полный путь к репозиторию с датасетами вида: "C:\\...\\data"
data_name	Название файла .csv с датасетом вида: "dataset"

```
fstream fin;
fin.open(file_path + "\\\" + data_name + ".csv", ios::in);
string line;
vector<vector<string>> parsedCsv;
if (fin.fail()) {
    cout << "NOT OPEN";
}
while (getline(fin, line)) {
    stringstream lineStream(line);
    string cell;
    vector<string> parsedRow;
    while (getline(lineStream, cell, ',')) {
        parsedRow.push_back(cell);
    }

    parsedCsv.push_back(parsedRow);
}
fin.close();
dataset_ = parsedCsv;

for (int i = 1; i < dataset_.size(); i++) {
    vector<double> cell;
    for (int j = 0; j < dataset_[i].size(); j++) {
        if (j != dataset_[i].size() - 1) {
            cell.push_back(stod(dataset_[i][j]));
        } else if (j == dataset_[i].size() - 1) {
            y_.push_back(stoi(dataset_[i][j]));
        }
    }
    X_.push_back(cell);
}
X_ = data_normalization(X_);
```

3.1.4 Данные класса

dataset_

```
vector<vector<string>> > DiabetesData::dataset_ [private]
```

Двумерный вектор типа double, содержащий в себе выборку

features_

```
vector<vector<double> > DiabetesData::features_ [private]
```

Двумерный вектор типа double, содержащий в себе выборку для использования на уже обученной модели

X_

```
vector<vector<double> > DiabetesData::X_ [private]
```

Двумерный вектор типа double, содержащий в себе нормализованную выборку

y_

```
vector<int> DiabetesData::y_ [private]
```

Вектор типа int, содержащий в себе итог(outcome)

Объявления и описания членов классов находятся в файлах:

- prj.lab/diabetes_data/include/diabetes_data/diabetes_data.hpp
- prj.lab/diabetes_data/diabetes_data.cpp

3.2 Класс LogisticRegression

Класс [LogisticRegression](#) предназначен для обучения модели на основе лог.регрессии

```
#include <regression.hpp>
```

Открытые члены

- [LogisticRegression](#) (const vector< vector< double > > &X, const vector< int > &y)
- vector< double > [fit](#) (int max_iter=100, double lr=0.1)
- double [loss](#) (vector< int > y, vector< vector< double > > z)

Открытые статические члены

- static vector< vector< double > > [logit](#) (vector< vector< double > > X, vector< double > w)
- static vector< vector< double > > [sigmoid](#) (vector< vector< double > > logits)
- static void [save_weights](#) (const vector< double > &weights)
- static vector< vector< double > > [predict_proba](#) (vector< vector< double > > feauters)
- static vector< int > [predict](#) (const vector< vector< double > > &feauters, double threshold=0.5)
- static int [model_accuracy](#) (vector< int > results, vector< int > y)
- static void [saveLossToCSV](#) (const vector< double > &losses)

Закрытые данные

- `vector< vector< double > > X_`
- `vector< int > y_`
- `vector< double > w_`
- `vector< double > losses_`
- `int max_iter_ = 0`
- `double lr_ = 0`

3.2.1 Подробное описание

Класс `LogisticRegression` предназначен для обучения модели на основе лог.регрессии

Класс `LogisticRegression` предоставляет возможность обучить модель, основанную на лог.регрессии, путем градиентного спуска по функции потерь с использованием R2 - регуляризации

3.2.2 Конструктор(ы)

`LogisticRegression()`

```
LogisticRegression::LogisticRegression (
    const vector< vector< double > > & X,
    const vector< int > & y )
```

Конструктор сохраняет поступившие данные и рандомит веса для дальнейшей работы

Аргументы

X	Двумерный вектор типа <code>double</code> , содержащий в себе нормализованную выборку
y	Вектор типа <code>int</code> , содержащий в себе итог(outcome)

```
X_ = X;
y_ = y;
for (int i = 0; i < X[0].size() + 1; i++) {
    double a = rand() % 1000;
    w_.push_back(a / 1000);
}
```

3.2.3 Методы

`fit()`

```
vector< double > LogisticRegression::fit (
    int max_iter = 100,
    double lr = 0.1 )
```

Функция, обучающая нашу модель по градиенту функции потерь логистической регрессии

Аргументы

max_iter	Переменная типа <code>int</code> , содержащая в себе кол-во итераций, которые будет совершать модель при обучении
----------	---

Предупреждения

Большие значение дают большую точность, но значительно увеличивают время обучения, для быстрого результата советуем использовать значение 10

Аргументы

lr	Переменная типа double, содержащая в себе коэффициент скорости обучения модели
----	--

Возвращает

Вектор типа double, содержащий в себе результаты функции потерь на каждой итерации модели при обучении

logit()

```
vector< vector< double > > LogisticRegression::logit (
    vector< vector< double > > X,
    vector< double > w ) [static]
```

Функция подсчитывает логиты всей выборки по текущим весам и возвращает их

Аргументы

X	Транспонированный двумерный вектор типа double, содержащий в себе выборку
w	Вектор весов типа double

Возвращает

Двумерный вектор логитов типа double

Формула по который мы вычисляем логиты:

$$\sum_{i=0}^n b + w_i * x_i$$

```
unsigned long long int m = X.size();
unsigned long long int n = X[0].size();
vector<vector<double>> logits(m, vector<double>(1, 0.0));
for (int i = 0; i < m; i++) {
    for (int j = 0; j < 1; j++) {
        for (int k = 0; k < n; k++) {
            logits[i][j] += X[i][k] * w[k];
        }
    }
}
return logits;
```

loss()

```
double LogisticRegression::loss (
    vector< int > y,
    vector< vector< double > > z )
```

Функция подсчитывает "функцию потерь" нашей логистической регрессии с использованием l2 - регуляризации, основываясь на y исходе(outcome) и полученных сигмоидах.

Аргументы

y	Вектор типа int, содержащий в себе итог(outcome) для нашей выборки
z	Двумерный вектор типа double, содержащий в себе сигмоиды для каждого логита

Возвращает

Число типа double показывающее текущее значение функции потерь

Формула по которой мы вычисляем функцию потерь:

$$\sum_{i=0}^n (y_i * \log(\sigma_i) + (1 - y_i) * \log(1 - \sigma_i)) + \sum_{i=0}^l w_i^2;$$

```
double loss = 0;
for (int i = 0; i < y.size(); i++) {
    loss += (y[i] * (log(z[i][0])) + (1 - y[i]) * log(1 - z[i][0]));
}
for (double i: w_)
    loss += pow(i, 2);

loss /= y.size();
loss *= -1;
return loss;
```

model_accuracy()

```
int LogisticRegression::model_accuracy (
    vector< int > results,
    vector< int > y ) [static]
```

Функция показывает точность модели в виде процентов

Аргументы

results	вектор предсказанных результатов тестовой выборки типа int
y	вектор действительных результатов тестовой выборки типа int

Возвращает

целое число типа int, показывающее точность модели

predict()

```
vector< int > LogisticRegression::predict (
    const vector< vector< double > > & feautres,
    double threshold = 0.5 ) [static]
```

Функция возвращает предсказания модели по предоставленной выборке

Аргументы

feauters	двумерный вектор типа double, содержащий в себе выборку
threshold	переменная типа double

Возвращает

вектор предсказаний типа int

predict_proba()

```
vector< vector< double > > LogisticRegression::predict_proba (
    vector< vector< double > > feauters ) [static]
```

Функция выдает сигмоиды по выборке с использованием наилучших весов, наша функция предсказаний

Аргументы

feauters	двумерный вектор типа double, содержащий в себе выборку
----------	---

Возвращает

двумерный вектор типа double, содержащий в себе сигмоиды для каждого логита нашей выборки

save_weights()

```
void LogisticRegression::save_weights (
    const vector< double > & weights ) [static]
```

Функция сохраняет наилучшие найденные веса в виде weights.txt файла, основываясь на функции потерь

Принцип сохранения весов в функции [fit\(\)](#):

```
f((!losses_.empty()) && (loss(y_, z) < *min_element(losses_.begin(), losses_.end())) {
    save_weights(w_);
}
losses_.push_back(loss(y_, z));
```

Аргументы

weights	двумерный вектор весов типа double
---------	------------------------------------

saveLossToCSV()

```
void LogisticRegression::saveLossToCSV (
    const vector< double > & losses ) [static]
```

Функция сохраняет "функции потерь" нашей модели на каждой итерации в файл losses_.csv вида loss,iteration:

1,0

0.80,1

0.5,2

...

n1, n2

Аргументы

losses	веткор потерь нашей модели на каждой итерации типа double
--------	---

sigmoid()

```
vector< vector< double > > LogisticRegression::sigmoid (
    vector< vector< double > > logits ) [static]
```

Функция подсчитывает сигмоиды по полученным логитам

Аргументы

logits	Двумерный вектор логитов типа double
--------	--------------------------------------

Возвращает

двумерный вектор типа double, содержащий в себе сигмоиды для каждого логита

Формула по который мы вычисляем логиты:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

```
vector<vector<double>> sigmoids;
for (int i = 0; i < logits.size(); i++) {
    vector<double> sigmoid;
    sigmoid.reserve(logits[0].size());
    for (int j = 0; j < logits[0].size(); j++)
        sigmoid.push_back(1 / (1 + exp(-logits[i][j])));
    sigmoids.push_back(sigmoid);
}
return sigmoids;
```

3.2.4 Данные класса

losses_

```
vector<double> LogisticRegression::losses_ [private]
```

Вектор типа double, содержащий в себе результаты функции потерь на каждой итерации модели при обучении

lr_

```
double LogisticRegression::lr_ = 0 [private]
```

Переменная типа double, содержащая в себе коэффициент скорости обучения модели

max_iter_

```
int LogisticRegression::max_iter_ = 0 [private]
```

Переменная типа int, содержащая в себе кол-во итераций, которые будет совершать модель при обучении

w_

```
vector<double> LogisticRegression::w_ [private]
```

Вектор типа double, содержащий в себе веса наней модели

X_

```
vector<vector<double> > LogisticRegression::X_ [private]
```

Двумерный вектор типа double, содержащий в себе нормализованную выборку

y_

```
vector<int> LogisticRegression::y_ [private]
```

Вектор типа int, содержащий в себе итог(outcome)

Объявления и описания членов классов находятся в файлах:

- [prj.lab/regression/include/regression/regression.hpp](#)
- [prj.lab/regression/regression.cpp](#)

3.3 Класс Plot

Класс [Plot](#) предназначен для создания .tex файла с информацией о модели

```
#include <plot.hpp>
```

Открытые члены

- [Plot](#) ()=default

Открытые статические члены

- static void [CreateLatexFile](#) (vector< int > results, vector< int > y, int max_iter=100)

3.3.1 Подробное описание

Класс [Plot](#) предназначен для создания .tex файла с информацией о модели

Класс [Plot](#) предоставляет возможность вывести данные о текущей модели в виде .tex файла, а именно: функция потерь и confusion matrix.

3.3.2 Конструктор(ы)

[Plot\(\)](#)

[Plot::Plot \(\)](#) [default]

Конструктор по умолчанию для создания объекта класса

3.3.3 Методы

[CreateLatexFile\(\)](#)

```
void Plot::CreateLatexFile (
    vector< int > results,
    vector< int > y,
    int max_iter = 100 ) [static]
```

Функция создает statistic.tex файл по результатам работы модели модели, содержащий в себе функцию потерь и confusion matrix

Аргументы

results	вектор типа int, содержащий в себе результаты предсказаний модели
y	вектор типа int, содержащий в себе действительный результат
max_iter	переменная типа int, содержащая в себе кол-во итераций, которая совершила модель при обучении (по умолчанию 100) Пример:

```
int YESaYES = 0;
int YESaNO = 0;
int NOaNO = 0;
int NOaYES = 0;
for (int i = 0; i < results.size(); i++) {
    if (results[i] == 1 && y[i] == 1) {
        YESaYES++;
    } else if (results[i] == 1 && y[i] == 0) {
        YESaNO++;
    } else if (results[i] == 0 && y[i] == 0) {
        NOaNO++;
    } else if (results[i] == 0 && y[i] == 1) {
        NOaYES++;
    }
}
const string &filename = "statistic.tex";
ofstream outputFile(filename);
if (outputFile.is_open()) {
    outputFile << "\\documentclass{article}\\n"
```

```

"\usepackage{pgfplots}\n"
"\usepackage{csvsimple}\n"
"\usepackage{pgfplotstable}\n"
"\usepackage{array}\n"
"\usepackage{graphicx}\n"
"\usepackage{multirow}\n"
"\usepackage{float} % Add the float package\n"
"\n"
"\newcommand{\MyBox}[2]{\n"
"\fbox{\lower0.75cm\n"
"\vbox to 1.7cm{\vfil\n"
"\hbox to 1.7cm{\hfil\parbox{0.4cm}{#1}\hfil}\n"
"\vfil}}\n"
"}%\n"
"}\n"
"\n"
"\begin{document}\n"
"\title{Model Statistic}\n"
"\maketitle\n"
"\n"
"\vspace{1cm}\n"
"\n"
"\begin{figure}[H] % Use the H specifier from the float package\n"
"\centering\n"
"\begin{tikzpicture}\n"
"\begin{axis}\n"
"\xlabel={Iters},\n"
"\ylabel={Log Loss},\n"
"\xtick=data,\n"
"\xticklabel={\pgfmathprintnumber[int detect]{\tick}},\n"
"\xmin=0, xmax=" + to_string(max_iter) + ",\n"
"\ymin=0, ymax=1,\n"
"\grid=both,\n"
"\width=14cm, height=8cm,\n"
"\samples=100\n"
"\n"
"\n"
"\pgfplotstableread[col sep=comma]{losses_.csv}\datatable\n"
"\addplot[blue] table[x index=1, y index=0] {\datatable};\n"
"\n"
"\end{axis}\n"
"\end{tikzpicture}\n"
"\caption{Loss changes every iteration}\n"
"\end{figure}\n"
"\n"
"\vspace{1cm}\n"
"\n"
"\begin{figure}[H] % Use the H specifier from the float package\n"
"\centering\n"
"\renewcommand{\arraystretch}{1.5}\n"
"\setlength{\tabcolsep}{0pt}\n"
"\begin{tabular}{c >{\bfseries}r @{\hspace{0.7em}}c @{\hspace{0.4em}}c @{\hspace{0.7em}}l\n"
"\multirow{10}{*}{\rotatebox{90}{\parbox{1.1cm}{\bfseries\centering actual\value}} & \n"
"& \multicolumn{2}{c}{\bfseries Prediction outcome} & \n"
"& \bfseries p & \bfseries n & \bfseries total \n"
"& p\$\$ & \MyBox{" + to_string(YESaYES) +  

"}{Positive} & \MyBox{" + to_string(NOaYES) +  

"}{Positive} & P\$\$ \n"
"& n\$\$ & \MyBox{" + to_string(YESaNO) + "}{Positive} & \MyBox{" + to_string(NOaNO) +  

"}{Positive} & N\$\$ \n"
"& total & P & N & \n"
"\end{tabular}\n"
"\caption{Confusion Matrix}\n"
"\end{figure}\n"
"\n"
"\end{document}";
} else {
cerr << "Unable to open file: " << filename << endl;
}

```

Объявления и описания членов классов находятся в файлах:

- prj.lab/plot/include/plot/plot.hpp
- prj.lab/plot/plot.cpp

4 Файлы

4.1 Файл prj.lab/diabetes_data/diabetes_data.cpp

```

#include <iostream>
#include <utility>

```

```
#include <vector>
#include <fstream>
#include <sstream>
#include <string>
#include <cmath>
#include <diabetes_data/diabetes_data.hpp>
```

4.2 Файл prj.lab/diabetes_data/include/diabetes_data/diabetes_data.hpp

```
#include <iostream>
#include <utility>
#include <vector>
#include <fstream>
#include <sstream>
#include <string>
#include <cmath>
```

Классы

- class [DiabetesData](#)

Класс [DiabetesData](#) предназначен для подготовки датасета

Макросы

- #define [DIABETES_AI_HPP_09052023DDAT](#)

4.2.1 Макросы

[DIABETES_AI_HPP_09052023DDAT](#)

```
#define DIABETES_AI_HPP_09052023DDAT
```

4.3 diabetes_data.hpp

[См. документацию.](#)

```
00001 //
00002 // Created by hedge on 13.05.2023.
00003 //
00004
00005 #pragma once
00006 #ifndef DIABETES_AI_HPP_09052023DDAT
00007 #define DIABETES_AI_HPP_09052023DDAT
00008
00009 #include <iostream>
00010 #include <utility>
00011 #include <vector>
00012 #include <fstream>
00013 #include <sstream>
00014 #include <string>
00015 #include <cmath>
00016
00017
00018 using namespace std;
00019
00024 class DiabetesData {
```



```

00025 public:
00029     DiabetesData() = default;
00036     explicit DiabetesData(const string &file_path, const string &data_name);
00037
00043     explicit DiabetesData(const vector<vector<double>> &features);
00049     void load_data_from_file(const string &file_path, const string &data_name);
00054     static vector<vector<double>> data_normalization(vector<vector<double>> X);
00055
00060     vector<vector<double>> get_X ();
00065     vector<int> get_y ();
00066
00067 private:
00071     vector<vector<double>> features_;
00075     vector<vector<string>> dataset_;
00079     vector<vector<double>> X_;
00083     vector<int> y_;
00084 };
00085
00086 #endif

```

4.4 Файл prj.lab/plot/include/plot/plot.hpp

```

#include <iostream>
#include <vector>
#include <fstream>
#include <string>

```

Классы

- class [Plot](#)

Класс [Plot](#) предназначен для создания .tex файла с информацией о модели

Макросы

- #define [DIABETES_AI_HPP_09052023PLOT](#)

4.4.1 Макросы

[DIABETES_AI_HPP_09052023PLOT](#)

```
#define DIABETES_AI_HPP_09052023PLOT
```

4.5 plot.hpp

[См. документацию.](#)

```

00001 //
00002 // Created by hedge on 09.05.2023.
00003 //
00004 #pragma once
00005 #ifndef DIABETES_AI_HPP_09052023PLOT
00006 #define DIABETES_AI_HPP_09052023PLOT
00007
00008 #include <iostream>
00009 #include <vector>
00010 #include <fstream>
00011 #include <string>
00012
00013
00014 using namespace std;
00015
00021 class Plot {
00022 public:
00026     Plot() = default;
00038     static void CreateLatexFile(vector<int> results, vector<int> y, int max_iter = 100);
00039 private:
00040 };
00041
00042 #endif

```

4.6 Файл prj.lab/plot/plot.cpp

```
#include <iostream>
#include <vector>
#include <fstream>
#include <string>
#include <plot/plot.hpp>
```

4.7 Файл prj.lab/regression/include/regression/regression.hpp

```
#include <iostream>
#include <utility>
#include <vector>
#include <fstream>
#include <sstream>
#include <string>
#include <cmath>
```

Классы

- class [LogisticRegression](#)

Класс [LogisticRegression](#) предназначен для обучения модели на основе лог.регрессии

Макросы

- #define [DIABETES_AI_HPP_09052023REG](#)

4.7.1 Макросы

[DIABETES_AI_HPP_09052023REG](#)

```
#define DIABETES_AI_HPP_09052023REG
```

4.8 regression.hpp

[См. документацию.](#)

```
00001 //
00002 // Created by hedge on 09.05.2023.
00003 //
00004
00005 #pragma once
00006 #ifndef DIABETES_AI_HPP_09052023REG
00007 #define DIABETES_AI_HPP_09052023REG
00008
00009 #include <iostream>
00010 #include <utility>
00011 #include <vector>
00012 #include <fstream>
00013 #include <sstream>
00014 #include <string>
00015 #include <cmath>
00016
00017 using namespace std;
00018
```

```

00023 class LogisticRegression {
00024 public:
00030     LogisticRegression(const vector<vector<double>> &X, const vector<int> &y);
00031
00043     static vector<vector<double>> logit(vector<vector<double>> X, vector<double> w);
00044
00055     static vector<vector<double>> sigmoid(vector<vector<double>> logits);
00056
00064     vector<double> fit(int max_iter = 100, double lr = 0.1);
00065
00077     double loss(vector<int> y, vector<vector<double>> z);
00078
00091     static void save_weights(const vector<double> &weights);
00092
00098     static vector<vector<double>> predict_proba(vector<vector<double>> feauters);
00099
00106     static vector<int> predict(const vector<vector<double>> &feauters, double threshold = 0.5);
00107
00114     static int model_accuracy(vector<int> results, vector<int> y);
00115
00133     static void saveLossToCSV(const vector<double> &losses);
00134
00135 private:
00139     vector<vector<double>> X_;
00143     vector<int> y_;
00147     vector<double> w_;
00151     vector<double> losses_;
00155     int max_iter_ = 0;
00161     double lr_ = 0;
00162 };
00163
00164 #endif

```

4.9 Файл prj.lab/regression/regression.cpp

```

#include <iostream>
#include <utility>
#include <vector>
#include <fstream>
#include <sstream>
#include <string>
#include <cmath>
#include <regression/regression.hpp>

```

4.10 Файл prj.test/main.cpp

```

#include <regression/regression.hpp>
#include <diabetes_data/diabetes_data.hpp>
#include <plot/plot.hpp>

```

Функции

- int main ()
Тестовое приложение библиотеки DiabetesAI.

4.10.1 Функции

main()

int main ()

Тестовое приложение библиотеки DiabetesAI.

В данной функции мы проверяем основные методы нашей библиотеки

Предметный указатель

CreateLatexFile

Plot, [13](#)

data_normalization

DiabetesData, [4](#)

dataset_

DiabetesData, [5](#)

DIABETES_AI_HPP_09052023DDAT

diabetes_data.hpp, [15](#)

DIABETES_AI_HPP_09052023PLOT

plot.hpp, [16](#)

DIABETES_AI_HPP_09052023REG

regression.hpp, [17](#)

diabetes_data.hpp

DIABETES_AI_HPP_09052023DDAT, [15](#)

DiabetesData, [2](#)

data_normalization, [4](#)

dataset_, [5](#)

DiabetesData, [3](#)

features_, [5](#)

get_X, [4](#)

get_y, [4](#)

load_data_from_file, [5](#)

X_, [6](#)

y_, [6](#)

features_

DiabetesData, [5](#)

fit

LogisticRegression, [7](#)

get_X

DiabetesData, [4](#)

get_y

DiabetesData, [4](#)

load_data_from_file

DiabetesData, [5](#)

LogisticRegression, [6](#)

fit, [7](#)

LogisticRegression, [7](#)

logit, [8](#)

loss, [8](#)

losses_, [11](#)

lr_, [11](#)

max_iter_, [12](#)

model_accuracy, [9](#)

predict, [9](#)

predict_proba, [10](#)

save_weights, [10](#)

saveLossToCSV, [10](#)

sigmoid, [11](#)

w_, [12](#)

X_, [12](#)

y_, [12](#)

logit

LogisticRegression, [8](#)

loss

LogisticRegression, [8](#)

losses_

LogisticRegression, [11](#)

lr_

LogisticRegression, [11](#)

main

main.cpp, [18](#)

main.cpp

main, [18](#)

max_iter_

LogisticRegression, [12](#)

model_accuracy

LogisticRegression, [9](#)

Plot, [12](#)

CreateLatexFile, [13](#)

Plot, [13](#)

plot.hpp

DIABETES_AI_HPP_09052023PLOT, [16](#)

predict

LogisticRegression, [9](#)

predict_proba

LogisticRegression, [10](#)

prj.lab/diabetes_data/diabetes_data.cpp, [14](#)

prj.lab/diabetes_data/include/diabetes_data/diabetes_data.hpp,
[15](#)

prj.lab/plot/include/plot/plot.hpp, [16](#)

prj.lab/plot/plot.cpp, [17](#)

prj.lab/regression/include/regression/regression.hpp,
[17](#)

prj.lab/regression/regression.cpp, [18](#)

prj.test/main.cpp, [18](#)

regression.hpp

DIABETES_AI_HPP_09052023REG, [17](#)

save_weights

LogisticRegression, [10](#)

saveLossToCSV

LogisticRegression, [10](#)

sigmoid

LogisticRegression, [11](#)

w_

LogisticRegression, [12](#)

X_

DiabetesData, [6](#)

LogisticRegression, [12](#)

y_

DiabetesData, [6](#)

LogisticRegression, [12](#)