$\begin{array}{c} {\rm Diabetes AI} \\ {\rm 1.0.0} \end{array}$

Создано системой Doxygen 1.9.7

1 Алфавитный указатель классов	1
1.1 Классы	1
2 Список файлов	3
2.1 Файлы	3
3 Классы	5
3.1 Класс DiabetesData	5
3.1.1 Подробное описание	5
3.1.2 Конструктор(ы)	6
3.1.2.1 DiabetesData() [1/3]	6
3.1.2.2 DiabetesData() [2/3]	6
$3.1.2.3~\mathrm{DiabetesData}()~\mathrm{_{[3/3]}}~\mathrm{_{\ldots}}$	6
3.1.3 Методы	6
3.1.3.1 data_normalization()	6
$3.1.3.2 \mathrm{get} \underline{\hspace{1em}} \mathrm{X}() \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots $	7
$3.1.3.3 \text{ get_y()}$	7
3.1.3.4 load_data_from_file()	7
3.1.4 Данные класса	8
3.1.4.1 dataset	8
$3.1.4.2~\mathrm{features}$	8
$3.1.4.3~{\rm X}_$	8
3.1.4.4 y	9
3.2 Класс LogisticRegression	9
3.2.1 Подробное описание	9
3.2.2 Конструктор(ы)	9
$3.2.2.1 \; \text{LogisticRegression}() \; \ldots \; $	9
3.2.3 Методы	10
$3.2.3.1~\mathrm{fit}()$	10
$3.2.3.2 \operatorname{logit}() $	10
$3.2.3.3~\mathrm{loss}()$	11
$3.2.3.4 \; \mathrm{model_accuracy}() \; \ldots \; \ldots \; \ldots \; \ldots \; \ldots \; \ldots \; \ldots$	12
$3.2.3.5 \; \mathrm{predict}()$	12
$3.2.3.6~\mathrm{predict_proba}()$	12
3.2.3.7 save_weights()	13
3.2.3.8 saveLossToCSV()	13
3.2.3.9 sigmoid()	13
3.2.4 Данные класса	14
3.2.4.1 losses	14
3.2.4.2 lr	14
$3.2.4.3~\mathrm{max_iter_}$	14
3.2.4.4 w	14
$egin{array}{cccccccccccccccccccccccccccccccccccc$	15
3.2.4.6 y	15

3.3 Класс Plot	15
3.3.1 Подробное описание	15
3.3.2 Конструктор(ы)	15
$3.3.2.1 \; \mathrm{Plot}() \; \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	15
3.3.3 Методы	15
3.3.3.1 CreateLatexFile()	15
4 Файлы	19
4.1 diabetes_data.hpp	19
4.2 plot.hpp	19
4.3 regression.hpp	20
4.4 Файл prj.test/libtest.cpp	20
4.4.1 Подробное описание	21
4.4.2 Функции	21
4.4.2.1 main()	21
Предметный указатель	23

Глава 1

Алфавитный указатель классов

1.1 Классы

Классы с их кратким описанием.

Diabete	$_{ m esData}$	
	Класс DiabetesData предназначен для подготовки датасета	5
Logistic	m cRegression	
	Класс LogisticRegression предназначен для обучения модели на основе лог.↔	
	регрессии	9
Plot		
	Класс Plot предназначен для создания .tex файла с информацией о модели	15

Алфа	витный	указатель	классов
TIJI WU	DELLIDIE	ynasaronb	Transcor

Глава 2

Список файлов

2.1 Файлы

Полный список документированных файлов.

$prj.lab/diabetes_data/include/diabetes_data/diabetes_data.hpp \\ \dots \dots$	19
$prj.lab/plot/include/plot/plot.hpp \ . \ . \ . \ . \ . \ . \ . \ . \ . \$	19
prj.lab/regression/include/regression/regression.hpp	2 0
$\operatorname{prj.test/libtest.cpp}$	
Тестовое приложение библиотеки DiabetesAI	2 0

1 Список файлов

Глава 3

Классы

3.1 Класс Diabetes Data

Kласс 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 $> y_{\perp}$

3.1.1 Подробное описание

Kласс 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)

3.1.2.1 DiabetesData() [1/3]

```
DiabetesData::DiabetesData ( ) [default]
```

Конструктор по умолчанию для создания объекта класса

3.1.2.2 DiabetesData() [2/3]

Конструктор принимает на вход полный путь к директории с датасетами и название датасета, после чего он вызывает функцию load data from file

Аргументы

$_{ m in}$	${ m file_path}$	Полный путь к репозиторию с датасетами вида: "C:\\\\data"
$_{ m in}$	${ m data_name}$	Название файла .csv с датасетои вида: "dataset"

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

3.1.2.3 DiabetesData() [3/3]

Конструктор принимает на вход выборку и нормализует её с помощью data normalization

Аргументы

in	features	двумерный вектор double состоящий из одной строчки 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

3.1.3.1 data normalization()

```
\label{eq:vector} $\operatorname{vector}<\operatorname{double}>>\operatorname{DiabetesData}::\operatorname{data}_{\operatorname{normalization}} ($\operatorname{vector}<\operatorname{vector}<\operatorname{double}>>X \ ) \quad [\operatorname{static}]
```

 Φ ункция принимает на вход выборку X_ и нормализует её методом z-масштабирования

3.1 Класс DiabetesData 7

Аргументы

X | Двумерный вектор типа double, содержащий в себе выборку

```
\begin{array}{l} vector\!<\!double\!>\; avarage;\\ \textbf{for}\; (int\; j=0;\; j<\; X[0].size();\; j++)\; \{\\ double\; summ=0; \end{array}
     for (auto &i: X) {
         summ += i[j];
     avarage.push_back(summ / X.size());
  \begin{array}{ll} vector < double > \ deviation; \\ for \ (int \ j = 0; \ j < X[0].size(); \ j++) \ \{ \\ double \ summ = 0; \end{array} 
     for (auto &i: X)
         \widehat{summ} += \widehat{pow}((i[j] - avarage[j]), 2);
      deviation.push\_back(sqrt(summ\ /\ (X.size()\ -\ 1)));
  \begin{array}{lll} & \textbf{for (int } j = 0; \ j < X[0].size(); j{+}{+}) \ \{ \\ & \textbf{for (auto \&i: X)} \ \{ \\ & i[j] = (i[j] - avarage[j]) \ / \ deviation[j]; \\ & \textbf{if (i[j] > 3 * deviation[j])} \ \{ \\ & i[j] = 0; \end{array} 
     }
 return X;
3.1.3.2 \text{ get } X()
vector < vector < double >> DiabetesData::get_X ( )
\Phiункция возвращающая приватный член X_{\_}
Возвращает
         двумерный вектор типа double
return X_;
3.1.3.3 \text{ get}_y()
vector< int > DiabetesData::get_y ( )
Функция возвращающая приватный член у
Возвращает
          вектор типа int
return y_;
3.1.3.4 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 у $_{_}$

Аргументы

file_path	Полный путь к репозиторию с датасетами вида: "C:\\\\data"
data_name	Название файла .csv с датасетои 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_[1].size(); j++) {
            if (j != dataset_[1].size() - 1) {
                cell.push_back(stod(dataset_[i][j]));
        } else if (j == dataset_[1].size() - 1) {
                y_.push_back(stoi(dataset_[i][j]));
        }
}
X_.push_back(cell);
}
X__ edata_normalization(X__);</pre>
```

3.1.4 Данные класса

3.1.4.1 dataset

```
{\tt vector}{<}{\tt vector}{<}{\tt string}{\gt} > {\tt DiabetesData}{\tt ::dataset}\_ \quad [{\tt private}]
```

Двумерный вектор типа double, содержащий в себе выборку

3.1.4.2 features_

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

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

```
3.1.4.3 X
```

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

Двумерный вектор типа double, содержащий в себе нормализованную выборку

```
3.1.4.4 y_
```

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

Вектор типа int, содержащий в себе итог(outcome)

Объявления и описания членов классов находятся в файлах:

- $\bullet \ prj.lab/diabetes_data/include/diabetes_data/diabetes_data.hpp$
- prj.lab/diabetes data/diabetes data.cpp

3.2 Класс LogisticRegression

Kласс 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

Kласс LogisticRegression предназначен для обучения модели на основе лог.регрессии

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

3.2.2 Конструктор(ы)

3.2.2.1 LogisticRegression()

```
\label{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 double, содержащий в себе нормализованную выборку
у	Вектор типа int, 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);} }
```

```
3.2.3.1 fit()  \label{eq:constraint}  \mbox{vector} < \mbox{double} > \mbox{LogisticRegression::fit (} \\ \mbox{int max\_iter} = 100,
```

double lr = 0.1)

Функция, обучающая нашу модель по градиенту функции потерь логистической регрессии

Аргументы

max_iter	Переменная типа int, 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

```
3.2.3.2 \quad 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:

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

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

vector < vector < double > > z)

Функция подсчитывает "функцию потерь" нашей логистической регрессии с использованием 12 - регуляризации, основываясь на у исходе(outcome) и полученных сигмоидах.

Аргументы

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

Возвращает

Число типа double показывающее текущее значение функции потерь

Формула по который мы вычисляем функцию потерь:

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

```
 \begin{aligned} & \text{double loss} = 0; \\ & \text{for (int } i = 0; i < y.size(); i++) \; \{ \\ & \text{loss} + = (y[i] * (\log(z[i][0])) + (1 - y[i]) * log(1 - z[i][0])); \\ & \text{for (double i: w_)} \\ & \text{loss} + = pow(i, 2); \\ & \text{loss} \neq y.size(); \\ & \text{loss} * = -1; \\ & \text{return loss}; \end{aligned}
```

3.2.3.4 model accuracy()

```
\label{eq:continuous} \begin{split} & \mathrm{int\ LogisticRegression::model\_accuracy\ (} \\ & \mathrm{vector} < \mathrm{int\ } > \mathrm{results}, \\ & \mathrm{vector} < \mathrm{int\ } > \mathrm{y\ }) \quad [\mathrm{static}] \end{split}
```

Функция показывает точность модели в виде процентов

Аргументы

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

Возвращает

целое число типа int, показывающее точность модели

3.2.3.5 predict()

```
\label{eq:const} \begin{split} \text{vector} < \text{int} > \text{LogisticRegression::predict (} \\ \text{const vector} < \text{vector} < \text{double} >> \& \text{ feauters,} \\ \text{double threshold} = 0.5 \;) \quad [\text{static}] \end{split}
```

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

3.2.3.6 predict proba()

```
\label{eq:vector} $\operatorname{vector}<\operatorname{double}>>\operatorname{LogisticRegression}::\operatorname{predict\_proba}\ ($\operatorname{vector}<\operatorname{vector}<\operatorname{double}>>\operatorname{feauters}\ ) \quad [\operatorname{static}]
```

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

Аргументы

C	
Lteauters	двумерный вектор типа double, содержащий в себе выборку
TCGGGCIB	двумерный вектор типа абабте, содержащий в себе выборку

Возвращает

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

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

3.2.3.8 saveLossToCSV()

```
\label{eq:construction} \begin{tabular}{ll} void LogisticRegression::saveLossToCSV ( \\ &const vector < double > \& losses ) & [static] \end{tabular}
```

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

3.2.3.9 sigmoid()

```
{\tt vector} {\tt < vector} {\tt < double >> LogisticRegression} :: {\tt 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:

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

```
\label{eq:vector} $$ \ensuremath{\mathsf{vector}} < \ensuremath{\mathsf{double}} > \ensuremath{\mathsf{sigmoids}}; \\ $$ \ensuremath{\mathsf{for}} \ (\inf \ i = 0; \ i < \ensuremath{\mathsf{logits.size}}(); \ i++) \ \{ \\ \ensuremath{\mathsf{vector}} < \ensuremath{\mathsf{double}} > \ensuremath{\mathsf{sigmoid.size}}()); \\ \ensuremath{\mathsf{sigmoid.pesh-bock}} \ (\inf \ j = 0; \ j < \ensuremath{\mathsf{logits}}[0]. \ensuremath{\mathsf{size}}(); \ j++) \\ \ensuremath{\mathsf{sigmoid.push\_back}} \ (1 + \exp(-\ensuremath{\mathsf{logits}}[i][j]))); \\ \ensuremath{\mathsf{sigmoids.push\_back}} \ (\inf \ j = 0; \ j < \ensuremath{\mathsf{logits}} \ (i); \ j++) \\ \ensuremath{\mathsf{sigmoids.push\_back}} \ (\inf \ j = 0; \ j < \ensuremath{\mathsf{logits}} \ (i); \ j++) \\ \ensuremath{\mathsf{sigmoids.push\_back}} \ (\inf \ j = 0; \ j < \ensuremath{\mathsf{logits}} \ (i); \ j++) \\ \ensuremath{\mathsf{logits}} \ (i); \ j++ \ensuremath{\mathsf{logits}} \ (i); \ j++ \ensuremath{\mathsf{logits}} \ (i); \ j++) \\ \ensuremath{\mathsf{logits}} \ (i); \ j++ \ensuremath{\mathsf{logits}} \ (i); \ (i);
```

3.2.4 Данные класса

3.2.4.1 losses

vector<double> LogisticRegression::losses_ [private]

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

```
3.2.4.2 lr
```

 $double\ LogisticRegression:: lr_ = 0 \quad [private]$

Переменная типа double, содержащая в себе коэффициент скорости обучения модели

```
3.2.4.3 max_iter_
```

 $int \ LogisticRegression::max_iter_ = 0 \quad [private]$

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

```
3.2.4.4 w_
```

vector<double> LogisticRegression::w [private]

Вектор типа double, содержащий в себе веса наней модели

3.3 Класс Plot

```
3.2.4.5 X
```

```
{\tt vector}{<}{\tt vector}{<}{\tt double}{\gt} > {\tt LogisticRegression}{::}{\tt X}\_ \quad [{\tt private}]
```

Двумерный вектор типа double, содержащий в себе нормализованную выборку

```
3.2.4.6 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 Kласс 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 (int max iter=100, vector< int > results={}, vector< int > y={})
```

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)

$3.3.2.1 \, \text{Plot}()$

```
Plot::Plot ( ) [default]
```

Конструктор по умолчанию для создания объекта класса

3.3.3 Методы

3.3.3.1 CreateLatexFile()

```
void Plot::CreateLatexFile ( int \ max\_iter = 100, \\ vector < int > results = \{\}, \\ vector < int > y = \{\} ) \ [static]
```

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

Аргументы

max_iter	переменная типа int, содержащая в себе кол-во итераций, которая совершила модель при обучении (по умолчанию 100)
results	вектор типа int, содержащий в себе результаты предсказаний модели (по умолчанию пустой)
У	вектор типа int, содержащий в себе действительный результат (по умолчанию пустой) 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);
 "\\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"
"\\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"
                                                \\centering\n
                                              \\begin{axis}[\n"
    xlabel={Iters},\n"
    ylabel={Log Loss},\n"
                                                           xtick=data,\n"
                                                          xticklabel={\\pgfmathprintnumber[int detect]{\\tick}},"
xmin=0, xmax=" + to_string(max_iter) + ",\n"
ymin=0, ymax=1,\n"
                                                           grid=both,\n'
                                                           width=14cm, height=8cm,\n"
                                                           samples=100\n"
                                                   \label{losses_csv} $$ \sup_{csv}\datable\n'' \wedge [blue] table[x index=1, y index=0] {\datable};\n'' } $$
                                                   \\end{axis}\n"
                                               \\end{tikzpicture}\n"
                                        " \\caption{Loss changes every iteration}\n"
"\\end{figure}\n"
                                        "\n"
                                       "\\v space\{1cm\}\n"
                                        "\\begin{figure}[H] \% Use the H specifier from the float package\n"
                                                \\centering\n
                                              \label{lem:limit} $$ \operatorname{l.5}^n' \simeq \operatorname{l.5}^n' \end{0.7} c @{\hspace{0.7} c @{\hspace{0.7} c @{\hspace{0.7} c @{\hspace{0.7} c }} \n'' \end{0.7} c \end{
```

3.3 Класс Plot

```
" & & \\bfseries p & \\bfseries n & \\bfseries total \\\\n"
" & p$'$ & \\MyBox{" + to_string(YESaYES) +
"}{Positive} & \\MyBox{" + to_string(NOaYES) +
"}{Positive} & P$'$ \\\[2.4em]\n"
" & n$'$ & \\MyBox{" + to_string(YESaNO) + "}{Positive} & \\MyBox{" + to_string(NOaNO) +
"}{Positive} & N$'$ \\\\n"
" & total & P & N &\n"
" \cnd{tabular}\n"
" \caption{Confusion Matrix}\n"
"\caption{Gonfusion 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

Глава 4

Файлы

4.1 diabetes_data.hpp

```
00001
            Created by hedge on 13.05.2023.
00003 /
00004
00005 #pragma once
00006 #ifndef DIABETES_AI_HPP_09052023DDAT
00007 #define DIABETES_AI_HPP_09052023DDAT
00009 #include <iostream>
00010 #include <utility>
00011 #include <vector>
00012 #include <fstream>
00013 #include <sstream>
00014 #include <string>
00015 #include <cmath>
00016
00017
00018 \ \mathrm{using} \ \mathrm{namespace} \ \mathrm{std};
00019
00024 class DiabetesData {
00025 public:
              DiabetesData() = default;
explicit DiabetesData(const string &file_path, const string &data_name);
00029
00036
00037
00043
               \begin{array}{l} {\rm explicit\ DiabetesData(const\ vector< vector< double*\ \&features);} \\ {\rm void\ load\_data\_from\_file(const\ string\ \&file\_path,\ const\ string\ \&data\_name);} \\ {\rm static\ vector< vector< double*\ data\_normalization(vector< vector< double*\ X);} \\ \end{array} 
00049
00054
00055
00060
              {\tt vector}{<}{\tt vector}{<}{\tt double}{\tt * \ \ get}\_X\ ();
00065
              vector<int> get_y ();
00066
00067 private:
              vector<vector<double» features_;
00071
              vector<vector<string» dataset_;
00075
00079
              vector<vector<double ** X_;
00083
              vector < int > y_{-};
00084 };
00085
00086 #endif
```

4.2 plot.hpp

```
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
```

20 Файлы

4.3 regression.hpp

```
00001
00002
          Created by hedge on 09.05.2023.
00003
00004
00005~\#\mathrm{pragma} once
00006 #pragma blackers AI_HPP_09052023REG
00007 #define DIABETES_AI_HPP_09052023REG
00008
00009 #include <iostream>
00010 #include <utility>
00011 #include <vector>
00012~\#\mathrm{include}~<\!\!\mathrm{fstream}\!\!>
00013 #include <sstream>
00014 \#include <string>
00015~\# include < cmath >
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 \\ 00056
          static vector<vector<double» sigmoid(vector<vector<double» logits);
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
          static\ vector{<}vector{<}double{"}" proba(vector{<}vector{<}double{"}" feauters);
00098
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
          static void saveLossToCSV(const vector<double> &losses);
00133
00134
00135 private:
00139
          vector<vector<double > X_;
00143
          vector<int> y_;
          vector < double > w
00147
00151
          vector < double > losses_;
          \begin{array}{l} \mathrm{int}\; \max\_\mathrm{iter}\_ = 0;\\ \mathrm{double}\; \mathrm{lr}\_ = 0; \end{array}
00155
00161
00162 };
00163
00164 #endif
```

4.4 Файл prj.test/libtest.cpp

Тестовое приложение библиотеки DiabetesAI.

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

Функции

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

4.4.1 Подробное описание

Тестовое приложение библиотеки DiabetesAI.

В данном приложении мы тестируем основные функции и методы библиотеки, а именно: обработка датасета с помощью класса DiabetesData, обучение модели с помощью класса LogisticRegression по встроенному датасету 1 и кол-вом итераций = 10, вывод функции потерь в консоль, прогнозирование диабета по встроенному датасету 2 и вывод статистики модели с помощью класса Plot

4.4.2 Функции

```
4.4.2.1 \quad main() void \ main ( \\ int \ argc, \\ char * argv[])
```

Аргументы

argc	Название .exe файла для запуска тестового приложения
argv	полный путь до папки с датасетами вида "C:\\data"

```
      Обрабатываем встроенный датасет 1

      DiabetesData a(argv[1],"dataset1");

      vector<vector<double> X1 = a.get_X();

      vector<int> y1 = a.get_y();

      LogisticRegression lg1(X1, y1);

      Bыводим функции потерь в консоль vector<double> losses = lg1.fit(10);

      for (double losse: losses)

      cout « losse « endl;

      Обрабатываем встроенный датасет 2

      DiabetesData b(argv[1],"dataset2");

      vector<vector<double> X = a.get_X();

      vector<int> y = a.get_y();

      Берем предсказания датасета 2 из обученной модели LogisticRegression lg2(X, y);

      vector<int> results = LogisticRegression::predict(X);

      Cоздаем statistic.tex файл

      Plot::CreateLatexFile(results, y, 10);
```

22 Файлы

Предметный указатель

Piot, 15	CreateLatexFile	LogisticRegression, 10
DiabetesData, 6	Plot, 15	loss
DiabetesData, 6 LogisticRegression, 14		LogisticRegression, 11
DiabetesData, 6 dataset DiabetesData, 8 DiabetesData, 8 DiabetesData, 8 DiabetesData, 8 DiabetesData, 6 dataset_, 8 DiabetesData, 6 dataset_, 8 DiabetesData, 6 dataset_, 8 DiabetesData, 6 fentures_, 8 get_Y, 7 load data from file, 7 X_, 8 Y_, 8 Plot, 15 CreateLatexFile, 15 Plot, 15 Predict DiabetesData, 8 fit LogisticRegression, 10 get_Y DiabetesData, 7 get_Y DiabetesData, 8 LogisticRegression, 14 T14 max iter14 model_accuracy, 11 predictproba, 12 save_weights_13 save_lossToCSV, 13 sigmoid, 13 w,14 X,14 X,14 Y,15	${ m data_normalization}$	9 9 1
Ir	DiabetesData, 6	
DiabetesData, 8	dataset	
DiabetesData, 5 data_normalization, 6 dataset_, 8 DiabetesData, 6 features_, 8 get_X, 7 get_Y, 7 DiabetesData, 8 fit LogisticRegression, 10 get_X DiabetesData, 7 get_y DiabetesData, 7 LogisticRegression, 10 get_X DiabetesData, 7 get_y DiabetesData, 7 get_y DiabetesData, 7 LogisticRegression, 12 prilab/plot/include/plot/plot.hpp, 19 prilab/regression/regression/regression/regression/regression/regression.hpp, 20 save_weights LogisticRegression, 13 saveLossToCSV LogisticRegression, 13 sigmoid LogisticRegression, 13 sigmoid LogisticRegression, 14 lr_, 14 max_iter_, 14 max_iter_, 14 model_accuracy DiabetesData, 8 LogisticRegression, 14 V_ DiabetesData, 8 LogisticRegression, 15 DiabetesData, 8 LogisticRegression, 15 DiabetesData, 8 LogisticRegression, 15		
data	DiabetesData, 5	
dataset		main
DiabetesData, 6 features, 8 getX, 7 gety, 7 load_data_from_file, 7 X, 8 y, 8 Plot, 15 CreateLatexFile, 15 Plot, 15 Plot, 15 Plot, 15 predict_ proba LogisticRegression, 12 prilab/plot/include/plot/plot.hpp, 19 prilab/regression/include/regression/regression.hpp, 20 prilab/regression/include/regression/regression.hpp, 20 prilab/regression, 13 sigmoid LogisticRegression, 9 logit, 10 LogisticRegression, 13 sigmoid LogisticRegression, 13 losses_, 14 lr, 14 maxiter, 14 modelaccuracy, 11 predict_, 12 predict_ proba, 12 save_weights, 13 save_LossFoCSV, 13 sigmoid, 13 w, 14 X, 14 Y, 15		
features , 8		± ± /
get_X, 7		
$ \begin{array}{c} \operatorname{get}_y, 7 \\ \operatorname{load}_\operatorname{data}_\operatorname{from}_\operatorname{file}, 7 \\ X_, 8 \\ y_, 8 \\ \end{array} $ $ \begin{array}{c} \operatorname{Plot}, 15 \\ \operatorname{CreateLatexFile}, 15 \\ \operatorname{Plot}, 15 \\ \end{array} $ $ \begin{array}{c} \operatorname{DiabetesData}, 8 \\ \operatorname{fit} \\ \operatorname{LogisticRegression}, 10 \\ \operatorname{get}_X \\ \operatorname{DiabetesData}, 7 \\ \operatorname{LogisticRegression}, 9 \\ \operatorname{ndin}, 21 \\ \operatorname{load}_\operatorname{data}_\operatorname{from}_\operatorname{file} \\ \operatorname{DiabetesData}, 7 \\ \operatorname{LogisticRegression}, 9 \\ \operatorname{fit}, 10 \\ \operatorname{LogisticRegression}, 9 \\ \operatorname{logist}, 10 \\ \operatorname{loss}, 11 \\ \operatorname{loses}_, 14 \\ \operatorname{lr}_, 14 \\ \operatorname{max}_\operatorname{iter}_, 14 \\ \operatorname{mode}_\operatorname{accuracy}, 11 \\ \operatorname{predict}_\operatorname{proba}, 12 \\ \operatorname{save} \\ \operatorname{weights}, 13 \\ \operatorname{save} \operatorname{LogisticRegression}, 14 \\ \operatorname{LogisticRegression}, 13 \\ \operatorname{LogisticRegression}, 14 \\ \operatorname{LogisticRegression}, 15 \\ LogisticRe$		
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		
Y_, 8		10810101061001011, 11
CreateLatexFile, 15 Plot, 15 Plot, 15 Plot, 15 Plot, 15 Plot, 15 Plot, 15 Predict DiabetesData, 8 fit LogisticRegression, 10 LogisticRegression, 10 LogisticRegression, 10 LogisticRegression, 12 prilab/diabetes_data/include/diabetes_data/diabetes_data.hpp, 19 prj.lab/plot/include/plot/plot.hpp, 19 prj.lab/regression/include/regression/regression.hpp, 20 prj.test/libtest.cpp, 20 libtest.cpp main, 21 load_data_from_file DiabetesData, 7 LogisticRegression, 9 fit, 10 LogisticRegression, 9 fit, 10 LogisticRegression, 9 logit, 10 loss, 11 losses_, 14 lr_, 14 max_iter_, 14 model_accuracy, 11 predict, 12 predict_proba, 12 save_weights, 13 save_lossion, 14 LogisticRegression, 15 LogisticRegression, 15 LogisticRegression, 15 LogisticRegression, 15 LogisticRegression, 15 LogisticRegression, 15		Plot. 15
Flot, 15 features_		
features	y_, o	
DiabetesData, 8 fit	features	
fit LogisticRegression, 10 LogisticRegression, 12 pri_lab/diabetes_data/include/diabetes_data.hpp, get_X DiabetesData, 7 get_y DiabetesData, 7 get_y DiabetesData, 7 get_y DiabetesData, 7 libtest.cpp main, 21 load_data_from_file DiabetesData, 7 LogisticRegression, 9 logit, 10 LogisticRegression, 9 logit, 10 loss, 11 losses_, 14 lr_, 14 max_iter_, 14 model_accuracy, 11 predict, 12 predict_proba LogisticRegression, 13 saveLossToCSV LogisticRegression, 14 lr_predict, 12 predict_proba LogisticRegression, 14 lr_predict, 12 predict_proba LogisticRegression, 14 logister_proba LogisticRegression, 14 losses_data/diabetes_data/diabetes_data.hpp, pri_lab/plot/include/plot/plot.hpp, 19 pri_lab/plot/include/plot.hpp, 19 pri_lab/plot/include/plot/plot.hpp, 19 pri_lab/plot/include/plot/plot.hpp, 19 pri_lab/plot/include/plot/plot.hpp, 19 pr		•
LogisticRegression, 10 get_X		9 9 1
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$		
get_X DiabetesData, 7 get_y DiabetesData, 7 get_y DiabetesData, 7 libtest.cpp main, 21 load_data_from_file DiabetesData, 7 LogisticRegression, 9 fit, 10 LogisticRegression, 9 logit, 10 loss, 11 losses_, 14 lr_, 14 max_iter_, 14 model_accuracy, 11 predict, 12 predict_proba, 12 save_weights, 13 saveLossToCSV, 13 sigmoid, 13 w_, 14 X_, 14 X_, 14 Y_, 15	Logisticitegi ession, 10	9 9 1
DiabetesData, 7 get_y DiabetesData, 7 get_y DiabetesData, 7 get_y DiabetesData, 7 get_y DiabetesData, 7 libtest.cpp main, 21 load_data_from_file DiabetesData, 7 LogisticRegression, 9 logit, 10 LogisticRegression, 9 logit, 10 loss, 11 losses, 14 lr, 14 max_iter, 14 model_accuracy, 11 predict, 12 predict_proba, 12 save_weights, 13 saveLossToCSV LogisticRegression, 13 LogisticRegression, 13 LogisticRegression, 13 LogisticRegression, 13 w LogisticRegression, 13 LogisticRegression, 13 LogisticRegression, 14 LogisticRegression, 15 LogisticRegression, 15 LogisticRegression, 15	oret X	
get_y	- -	
DiabetesData, 7 20		
Dibtest.cpp		
libtest.cpp main, 21 save_weights LogisticRegression, 13 SaveLossToCSV LogisticRegression, 9 LogisticRegression, 13 sigmoid LogisticRegression, 9 logit, 10 loss, 11 w_ logisticRegression, 14 lr_, 14 max_iter_, 14 model_accuracy, 11 predict, 12 predict_proba, 12 save_weights, 13 saveLossToCSV, 13 sigmoid, 13 w_, 14 X_, 14 X_{-} , 15	DianetesData, 1	
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	lihtest con	prj.test/libtest.cpp, 20
$\begin{array}{llllllllllllllllllllllllllllllllllll$		aorro mai abta
DiabetesData, 7		= 9
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$		
$\begin{array}{cccccccccccccccccccccccccccccccccccc$		
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	= -	
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	·	
$\begin{array}{cccccccccccccccccccccccccccccccccccc$		LogisticRegression, 13
$\begin{array}{cccccccccccccccccccccccccccccccccccc$		
$\begin{array}{cccccccccccccccccccccccccccccccccccc$		WIi_Di14
$\begin{array}{llllllllllllllllllllllllllllllllllll$	10SSes_, 14	Logistic negression, 14
$\begin{array}{llllllllllllllllllllllllllllllllllll$		V
predict, 12 LogisticRegression, 14 predict_proba, 12 save_weights, 13		NighatasData 8
predict_proba, 12 save_weights, 13 saveLossToCSV, 13 sigmoid, 13 w, 14 X, 14 y, 15		
$\begin{array}{cccccccccccccccccccccccccccccccccccc$		Logisticitegiession, 14
$\begin{array}{cccccccccccccccccccccccccccccccccccc$		V
sigmoid, 13 LogisticRegression, 15 w_{\perp} , 14 X_{\perp} , 14 y_{\perp} , 15		– DiabetesData, 8
w_, 14 X_, 14 y_, 15		
$egin{array}{c} { m X}, 14 \ { m y}, 15 \end{array}$	_	
$\mathrm{y}_{_},15$		
logit		
	logit	