МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЕЧСКИЙ УНИВЕРСИТЕТ «МИСИС»

Институт информационных технологий и компьютерных наук

Кафедра инженерной кибернетики

Курсовая работа

по дисциплине
«Объектно-ориентированное программирование»
на тему:

«БИБЛИОТЕКА ДЛЯ ДИАГНОСТИРОВАНИЯ ДИАБЕТА МЕТОДОМ ЛОГИСТИЧЕСКОЙ РЕГРЕССИИ»

Выполнил:
студент 1-го курса,
гр. БПМ-22-4 Оркин.Р.Р.

Проверил:
доцент, к.т.н. Полевой Д.В.

Москва 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель данной работы - создать библиотеку, которая даст возможность обучить модель логистической регрессии на нахождение заболевания диабетом у человека.

Функциональные требования:

- Загрузка данных: Возможность загрузки данных о диабете из файлового формата.
- Предварительная обработка данных: Функции для очистки и масштабирования данных.
- Обучение модели: Возможность обучения модели логистической регрессии на основе загруженных данных.
- Оценка модели: Предоставление метрик для оценки качества модели.
- Прогнозирование: Использование обученной модели для прогнозирования результатов на новых данных

1.1 Пользовательское описание

Проект DiabetesAI представляет собой библиотеку, специализированную на реализации и применении модели логистической регрессии для анализа данных, связанных с диабетом. Логистическая регрессия является статистической моделью, используемой для прогнозирования бинарных или категориальных результатов на основе входных переменных.

Основные возможности и функции библиотеки DiabetesAI для логистической регрессии включают:

1. Подготовка данных: Библиотека позволяет загружать и предварительно обрабатывать данные, связанные с диабетом, чтобы они были пригодны для применения логистической регрессии. Это может включать очистку данных, масштабирование, преобразование переменных и другие операции.
2. Обучение модели: DiabetesAI предоставляет функции для обучения логистической регрессии на основе подготовленных данных. Это включает определение целевой переменной, выбор соответствующих

признаков, настройку параметров модели и выполнение процесса обучения.

3. Оценка модели: Библиотека позволяет оценивать качество и эффективность обученной модели логистической регрессии с помощью различных метрик и методов, таких как точность, полнота, F-мера и кривая ROC.
4. Применение модели: DiabetesAI предоставляет возможность использовать обученную модель логистической регрессии для прогнозирования результатов на новых данных. Вы можете передавать входные данные в модель и получать предсказания, основанные на обученных весах и параметрах модели.

Проект DiabetesAI предназначен для разработчиков и исследователей, которым требуется эффективный инструмент для реализации и применения логистической регрессии в контексте анализа данных, связанных с диабетом. Библиотека предоставляет удобный и гибкий интерфейс, а также функции, необходимые для загрузки, подготовки, обучения и оценки модели логистической регрессии на основе данных о диабете.

1.1 Пользовательская документация

Список файлов

Объявления и описания членов классов находятся в файлах.

-prjcw.lab/diabetes_data/include/diabetes_data/diabetes_data.hpp
-prjcw.lab/plot/include/plot/plot.hpp
-prjcw.lab/regression/include/regression/regression.hpp
-prjcw.test/libtest.cpp (Тестовое приложение библиотеки DiabetesAI)

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_

Конструктор(ы)

DiabetesData::DiabetesData ()[default]

Конструктор по умолчанию для создания объекта класса

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

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 строчки и n столбцов (n = кол-ву столбцов датасета - 1)

Методы

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

Функция возвращающая приватный член X_

Возвращает

двумерный вектор типа double

vector< int > DiabetesData::get_y ()

Функция возвращающая приватный член y_

Возвращает

вектор типа int

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 датасетои вида: "dataset"

Данные класса

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

Двумерный вектор типа double, содержащий в себе выборку

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

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

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

Двумерный вектор типа double, содержащий в себе **нормализованную** выборку

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

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

Конструктор(ы)

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
y	Вектор типа int, содержащий в себе итог(outcome)

Методы

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 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

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

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

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

vector< int > LogisticRegression::predict (const vector< vector< double > > & feauters*, 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

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

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

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

Аргументы

weights	двумерный вектор весов типа double

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

Функция сохраняет "функции потерь" нашей модели на каждой итерации в файл losses_.csv

Аргументы

losses	веткор потерь нашей модели на каждой итерации типа double

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

Данные класса

vector<double> LogisticRegression::losses_[private]

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

double LogisticRegression::lr_ = 0[private]

Переменная типа double, содержащая в себе коэффициент скорости обучения модели

int LogisticRegression::max_iter_ = 0[private]

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

vector<double> LogisticRegression::w_[private]

Вектор типа double, содержащий в себе веса наней модели

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

Двумерный вектор типа double, содержащий в себе **нормализованную** выборку

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

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 (int max_iter=100, vector< int > results={}, vector< int > y={}) [static]

Конструктор(ы)

Plot::Plot ()[default]

Конструктор по умолчанию для создания объекта класса

Методы

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)
y	вектор типа int, содержащий в себе действительный результат (по умолчанию пустой) Пример:

Объявления и описания членов классов находятся в файлах:

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

1.2 Техническое описание

1.2.1 Общая информация

Основной функционал библиотеки доступен через три класса: DiabetesData, LogisticRegression, Plot.
– Библиотека написана на языке C++;
– Пользовательский интерфейс отсутствует, библиотека предоставляет только программный интерфейс;
– В случае некорректного ввода данных пользователь получает уведомление об ошибке.
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).

Класс LogisticRegression предоставляет возможность обучить модель, основанную на лог.регрессии, путем градиентного спуска по функции потерь с использованием L2 – регуляризации.

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

1.2.2 Принцип работы логистической регрессии

Логистическая регрессия — это статистический метод машинного обучения, используемый для решения задач классификации. Она основана на использовании логистической функции (сигмоиды) для моделирования вероятности принадлежности объекта к определенному классу.

Процесс работы логистической регрессии включает следующие шаги:

1. Сигмоидная функция: Логистическая регрессия использует сигмоидную функцию, которая преобразует входные значения в диапазоне от 0 до 1.

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

Формула 1

Формула сигмоидной функции на Рисунок 1.

2. Линейная комбинация: Логистическая регрессия осуществляет линейную

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

Формула 2

комбинацию входных признаков с их соответствующими весами. Формула линейной комбинации на Рисунок 2, где w - веса признаков, x - значения признаков, n - количество признаков, b - свободный член (смещение).

3. Обучение модели: Цель логистической регрессии - найти оптимальные значения весов (w) путем минимизации функции потерь. Обычно используется логарифмическая функция потерь (log loss), которая измеряет разницу между предсказанной вероятностью и фактической меткой класса.
4. Градиентный спуск: Для нахождения оптимальных значений весов используется метод градиентного спуска. Он обновляет значения весов в направлении наискорейшего убывания градиента функции потерь с L2 регуляризацией (Рисунок 3). Процесс обновления весов повторяется до достижения сходимости.

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

Формула 3

5. Прогнозирование: После обучения модели с оптимальными значениями весов, можно применить ее для прогнозирования классов новых объектов. Предсказанная вероятность может быть преобразована в классификацию, выбирая пороговое значение.

1.3 Инструкция по установке

Для работы библиотеки DiabetesAI и генерации документации необходимы – системы генерации Doxygen и дистрибутива TeX.

1. Скачайте исходный код библиотеки из репозитория проекта по ссылке https://github.com/avanturer/misis_homework_private
2. Откройте терминал или командную строку и перейдите в папку проекта orkin_r_r
3. Создайте директорию build для сборки проекта, выполнив команду: *mkdir build*
4. Перейдите в директорию build, выполнив команду: *cd build*
5. Создайте файлы проекта Cmake, выполнив команду: *cmake -DCMAKE_TOOLCHAIN_FILE="путь\до\vcpkg.cmake" ..*
6. Соберите проект, выполнив команду:

cmake --build . --config Release

7. Интсаллируйте проект в выбранную вами папку, выполнив команду: *cmake --install . --config Release --prefix "путь\до\папки"*
8. После успешной сборки проекта, вы можете запустить тестовые приложения в папке bin, а также посмотреть latex или html-документацию в папке docs, которые появились на одном уровне

1.4 Инструкция по тестированию

В приложении libtest.exe 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.

Для тестирования используются csv-файлы, которые находятся по пути *orkin_r_r\prj_cw\prjcw.lab\data*:

- *dataset.csv* содержит полный датасет с параметрами и исходом
- *dataset1.csv* содержит 80% датасета *dataset.csv* с параметрами и исходом
- *dataset2.csv* содержит 20% датасета *dataset.csv* с параметрами и исходом

Инструкция по тестированию приложения с использованием csv файлов:

1. Откройте папку bin с исполняемыми файлами *libtest.exe*
2. Через командную консоль запустите тестовое приложение и передайте путь к csv файлам (папке data) в качестве аргумента консольной строки: *libtest.exe ..\data*

3. После успешной передачи, через несколько секунд (или минут) в командной строке будут выведены функции потерь при каждой итерации, а после в рабочей директории появится statistic.tex файл.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

https://habr.com/ru/companies/skillfactory/articles/714244
https://www.mql5.com/ru/articles/10626