# Код на проекта

Проекта използва MVC (Model – View – Controller) архитектура, съответно нейният код е разпределен в различни папки. В папка test са разположени unit и functional тестовете. В папка node\_modules са всички библиотеки и framework-ци използвани от проекта, както и за нуждите на NodeJS.

## Основна папка

### Index.js

/\*\*

\* index.js

\* Start point of the program.

\*

\* @version 0.1

\* @author Plamen Nakov

\* @updated 2017-04-21

\*

\*/

//Express Documentation: https://expressjs.com/en/api.html

const EXPRESS = require('express');

const APP = EXPRESS();

process.on('uncaughtException', (err) => {

console.log('uncaughtException:\n' + err);

});

APP.use(require('./controllers'));

APP.listen(3000, () => {

console.log('Running http server on port 3000...');

});

module.exports = APP;

## Папка models

### matrix\_utils.js

/\*\*

\* math\_utils.js

\* Validates that matrix and its content.

\* Applies multiple regression and calculates coefficients.

\*

\* @version 0.1

\* @author Plamen Nakov

\* @updated 2017-04-21

\*

\*/

const MATH = require('mathjs');

exports.createMatrix = function createMatrix(args) {

var rows = args.rows;

var columns = args.columns;

var valuesArray = args.valuesArray;

var isDependent = args.isDependent;

//validate input matrix data

return new Promise(

(resolve, reject) => {

var fieldCounter = 0;

for(var option in args) {

++fieldCounter;

}

if(fieldCounter != 4) {

return reject('Invalid number of data set fields.');

}

if (rows < 1 && columns > 0) {

return reject('Invalid values for rows.');

}

if (rows > 0 && columns < 1) {

return reject('Invalid values for columns.');

}

if (valuesArray.length != rows \* columns) {

return reject('Invalid number of values in array.');

}

valuesArray.forEach(function (element) {

if (isNaN(element)) {

return reject('Invalid type of value in array.');

}

}, this);

var chunkedArray = [];

var chunk = columns;

if (isDependent) {

for (i = 0; i < valuesArray.length; i += chunk) {

tempArray = valuesArray.slice(i, i + chunk);

chunkedArray.push(tempArray);

}

} else {

for (i = 0; i < valuesArray.length; i += chunk) {

tempArray = valuesArray.slice(i, i + chunk);

tempArray.unshift(1);

chunkedArray.push(tempArray);

}

}

return resolve(MATH.matrix(chunkedArray));

});

};

//apply multiple regression logic and calculate coefficients

//MathJS Documentation: http://mathjs.org/docs/

exports.calculateCoefficients = function calculateCoefficients (indepndentMatrix, dependentMatrix) {

var transposedIndepndentMatrix = MATH.transpose(indepndentMatrix);

return MATH.multiply(

MATH.inv(MATH.multiply(transposedIndepndentMatrix, indepndentMatrix)),

MATH.multiply(transposedIndepndentMatrix, dependentMatrix));

};

## Папка controllers

### index.js

/\*\*

\* index.js

\* Define which controller to be used

\*

\* @version 0.1

\* @author Plamen Nakov

\* @updated 2017-04-21

\*

\*/

const ROUTER = require('express').Router();

ROUTER.use('/calculate', require('./calculate'));

ROUTER.use('\*', require('./not\_found'));

module.exports = ROUTER;

### calculate.js

/\*\*

\* calculate.js

\* Verifies HTTP request methods.

\* Create matrix and calculate its coeffiecients.

\*

\* @version 0.1

\* @author Plamen Nakov

\* @updated 2017-04-21

\*

\*/

//Express Documentation: https://expressjs.com/en/api.html

const ROUTER = require('express').Router();

const BODY\_PARSER = require('body-parser');

const MATRIX\_UTILS = require('../models/matrix\_utils');

ROUTER.use(BODY\_PARSER.urlencoded({

extended: false

}));

ROUTER.use(BODY\_PARSER.raw({ type: 'application/json' }));

//forbid use of GET method

ROUTER.get('/', (request, response) => {

return response.status(404).send({ error: 'Get method is not supported.' });

});

//validate POST request input

ROUTER.post('/', (request, response) => {

response.header('Content-Type', 'application/json');

var independentDataSet = null;

var dependentDataSet = null;

try {

if (request.header('Content-Type') != 'application/json') {

return response.status(400).send({ error: 'Content-Type:' + request.header('Content-Type') + ' is not supported.' });

}

requestBodyJson = JSON.parse(request.body.toString());

if (!requestBodyJson[0]) {

return response.status(400).send({ error: 'Empty request body.' });

}

independentDataSet = requestBodyJson[0].independentDataSet;

dependentDataSet = requestBodyJson[0].dependentDataSet;

if (!requestBodyJson[0].independentDataSet || !requestBodyJson[0].dependentDataSet) {

return response.status(400).send({ error: 'Undefined values for dataset' });

}

} catch (e) {

console.log('Error:\n' + e);

return response.status(400).send({ error: 'Verify your JSON object entries.' });

}

//lets try to calculate coefficients

MATRIX\_UTILS.createMatrix(independentDataSet)

.then(

indepndentMatrix => {

MATRIX\_UTILS.createMatrix(dependentDataSet)

.then(

dependentMatrix => {

var coefficients = MATRIX\_UTILS.calculateCoefficients(indepndentMatrix, dependentMatrix);

return response.status(200).send({ result: coefficients.\_data });

})

.catch(

failure => {

return response.status(400).send({ error: failure });

})

})

.catch(

failure => {

return response.status(400).send({ error: failure });

})

});

module.exports = ROUTER;

### not\_found.js

/\*\*

\* not\_found.js

\*

\* @version 0.1

\* @author Plamen Nakov

\* @updated 2017-04-21

\*

\*/

const ROUTER = require('express').Router();

const BODY\_PARSER = require('body-parser');

const MATRIX\_UTILS = require('../models/matrix\_utils');

ROUTER.use(BODY\_PARSER.urlencoded({

extended: true

}));

ROUTER.get('\*', (request, response) => {

return response.sendStatus(404);

});

ROUTER.post('\*', (request, response) => {

return response.sendStatus(404);

});

module.exports = ROUTER;

## Папка test

### matrix\_api.js

/\*\*

\* matrix\_api.js

\* Integrational and functional testing.

\*

\* @version 0.1

\* @author Plamen Nakov

\* @updated 2017-04-21

\*

\*/

//Chai Documentation: http://chaijs.com/guide/

const CHAI = require('chai');

const CHAI\_HTTP = require('chai-http');

const APP = require('../index');

CHAI.use(CHAI\_HTTP);

const expect = CHAI.expect;

const should = CHAI.should;

describe('Integration/functional tests:', function () {

var malformedBody;

var properBody;

it('Invalid route request.', (done) => {

CHAI.request(APP)

.post('/')

.send({})

.end((err, res) => {

expect(res).to.be.status(404);

done();

});

});

it('Invalid request header.', (done) => {

CHAI.request(APP)

.post('/calculate')

.set('content-type', 'application/x-www-form-urlencoded')

.send({})

.end((err, res) => {

expect(res).to.be.status(400);

expect(JSON.stringify(res.body.error)).to.equal('"Content-Type:application/x-www-form-urlencoded is not supported."');

done();

});

});

it('Empty body.', (done) => {

CHAI.request(APP)

.post('/calculate')

.send({})

.end((err, res) => {

expect(res.body.error).to.be.equal('Empty request body.');

expect(res).to.be.status(400);

done();

});

});

it('Returns JSON with the calculated coefficients.', (done) => {

properBody = [

{

"independentDataSet":

{

"rows": 6,

"columns": 3,

"valuesArray": [345, 65, 23, 168, 18, 18, 94, 0, 0, 187, 185, 98, 621, 87, 10, 255, 0, 0],

"isDependent": false

},

"dependentDataSet": {

"rows": 6,

"columns": 1,

"valuesArray": [31.4, 14.6, 6.4, 28.3, 42.1, 15.3],

"isDependent": true

}

}

];

var expectedResult = {

result:

[[0.5664574696019713],

[0.06532925469423256],

[0.008718736194584409],

[0.15104864761034875]]

};

CHAI.request(APP)

.post('/calculate')

.send(properBody)

.end((err, res) => {

expect(err).to.null;

expect(res).to.be.status(200);

expect(JSON.stringify(res.body)).to.equal(JSON.stringify(expectedResult));

done();

});

});

it('Invalid value in JSON "row" field.', (done) => {

malformedBody = [

{

"independentDataSet":

{

"rows": 0,

"columns": 3,

"valuesArray": [345, 65, 23, 168, 18, 18, 94, 0, 0, 187, 185, 98, 621, 87, 10, 255, 0, 0],

"isDependent": false

},

"dependentDataSet": {

"rows": 6,

"columns": 1,

"valuesArray": [31.4, 14.6, 6.4, 28.3, 42.1, 15.3],

"isDependent": true

}

}

];

CHAI.request(APP)

.post('/calculate')

.send(malformedBody)

.end((err, res) => {

expect(res.body.error).to.be.equal('Invalid values for rows.');

expect(res).to.be.status(400);

done();

})

});

it('Invalid value in JSON "column" field.', (done) => {

malformedBody = [

{

"independentDataSet":

{

"rows": 6,

"columns": 0,

"valuesArray": [345, 65, 23, 168, 18, 18, 94, 0, 0, 187, 185, 98, 621, 87, 10, 255, 0, 0],

"isDependent": false

},

"dependentDataSet": {

"rows": 6,

"columns": -1,

"valuesArray": [31.4, 14.6, 6.4, 28.3, 42.1, 15.3],

"isDependent": true

}

}

];

CHAI.request(APP)

.post('/calculate')

.send(malformedBody)

.end((err, res) => {

expect(res.body.error).to.be.equal('Invalid values for columns.');

expect(res).to.be.status(400);

done();

})

});

it('Invalid JSON fields for data sets.', (done) => {

malformedBody = [

{

"independentDataSet":

{

"rows": 6,

"columns": 3,

"valuesArray": [345, 65, 23, 168, 18, 18, 94, 0, 0, 187, 185, 98, 621, 87, 10, 255, 0, 0],

"dummy": 'test',

"isDependent": false

},

"dependentDataSet": {

"rows": 6,

"columns": 1,

"valuesArray": [31.4, 14.6, 6.4, 28.3, 42.1, 15.3],

"isDependent": true

}

}

];

CHAI.request(APP)

.post('/calculate')

.send(malformedBody)

.end((err, res) => {

expect(res.body.error).to.be.equal('Invalid number of data set fields.');

expect(res).to.be.status(400);

done();

})

});

it('Invalid number of elements in "valuesArray".', (done) => {

malformedBody = [

{

"independentDataSet":

{

"rows": 6,

"columns": 3,

"valuesArray": [345, 65, 23, 168, 18, 18, 94, 0, 0, 187, 185, 98, 621, 87, 10, 255, 0, 0],

"isDependent": false

},

"dependentDataSet": {

"rows": 6,

"columns": 1,

"valuesArray": [31.4, 14.6, 6.4, 28.3, 42.1, 15.3, 1],

"isDependent": true

}

}

];

CHAI.request(APP)

.post('/calculate')

.send(malformedBody)

.end((err, res) => {

expect(res.body.error).to.be.equal('Invalid number of values in array.');

expect(res).to.be.status(400);

done();

})

});

it('Invalid values in "valuesArray" field.', (done) => {

malformedBody = [

{

"independentDataSet":

{

"rows": 6,

"columns": 3,

"valuesArray": [345, 65, 23, 168, 18, 18, 94, 0, 0, 187, 185, 98, 621, 87, 'a', 255, 0, 0],

"isDependent": false

},

"dependentDataSet": {

"rows": 6,

"columns": 1,

"valuesArray": [31.4, 14.6, 6.4, 28.3, 42.1, 15.3],

"isDependent": true

}

}

];

CHAI.request(APP)

.post('/calculate')

.send(malformedBody)

.end((err, res) => {

expect(res.body.error).to.be.equal('Invalid type of value in array.');

expect(res).to.be.status(400);

done();

})

});

});

### matrix\_utils.js

/\*\*

\* matrix\_utils.js

\* Unit testing.

\*

\* @version 0.1

\* @author Plamen Nakov

\* @updated 2017-04-21

\*

\*/

//Chai Documentation: http://chaijs.com/guide/

//MathJS Documentation: http://mathjs.org/docs/

const chai = require('chai');

const chaiAsPromised = require('chai-as-promised');

const matrixUtils = require('../models/matrix\_utils');

const math = require('mathjs');

chai.use(chaiAsPromised);

const expect = chai.expect;

const should = chai.should;

describe('Unit tests:', function () {

var defaultRows = 6;

var defaultColumns = 3;

var defaultValuesArray = [345, 65, 23, 168, 18, 18, 94, 0, 0, 187, 185, 98, 621, 87, 10, 255, 0, 0];

var defaultIsDependent = false;

var dataSet = {

rows: defaultRows,

columns: defaultColumns,

valuesArray: defaultValuesArray,

isDependent: defaultIsDependent

};

var expectedResult;

it('Should create independent matrix successfully.', function () {

expectedResult = math.matrix(

[[1, 345, 65, 23],

[1, 168, 18, 18],

[1, 94, 0, 0],

[1, 187, 185, 98],

[1, 621, 87, 10],

[1, 255, 0, 0]]);

dataSet.rows = defaultRows;

dataSet.columns = defaultColumns;

dataSet.valuesArray = defaultValuesArray;

dataSet.isDependent = defaultIsDependent;

return expect(matrixUtils.createMatrix(dataSet)).to.eventually.become(expectedResult);

});

it('Should create dependent matrix successfully.', function () {

expectedResult = math.matrix(

[[345, 65, 23],

[168, 18, 18],

[94, 0, 0],

[187, 185, 98],

[621, 87, 10],

[255, 0, 0]]);

dataSet.rows = defaultRows;

dataSet.columns = defaultColumns;

dataSet.valuesArray = defaultValuesArray;

dataSet.isDependent = true;

return expect(matrixUtils.createMatrix(dataSet)).to.eventually.become(expectedResult);

});

it('Should fail invalid rows.', function () {

expectedResult = 'Invalid values for rows.';

dataSet.rows = 0;

dataSet.columns = 1;

dataSet.valuesArray = defaultValuesArray;

dataSet.isDependent = defaultIsDependent;

return expect(matrixUtils.createMatrix(dataSet)).to.eventually.rejectedWith(expectedResult);

});

it('Should fail invalid columns.', function () {

expectedResult = 'Invalid values for columns.';

dataSet.rows = 1;

dataSet.columns = 0;

dataSet.valuesArray = defaultValuesArray;

dataSet.isDependent = defaultIsDependent;

return expect(matrixUtils.createMatrix(dataSet)).to.eventually.rejectedWith(expectedResult);

});

it('Should fail smaller array size.', function () {

expectedResult = 'Invalid number of values in array.';

dataSet.rows = defaultRows;

dataSet.columns = defaultColumns;

dataSet.valuesArray = defaultValuesArray;

dataSet.isDependent = defaultIsDependent;

dataSet.valuesArray.length = 6 \* 3 - 1;

return expect(matrixUtils.createMatrix(dataSet)).to.eventually.rejectedWith(expectedResult);

});

it('Should fail larger array size.', function () {

expectedResult = 'Invalid number of values in array.';

dataSet.rows = defaultRows;

dataSet.columns = defaultColumns;

dataSet.valuesArray = defaultValuesArray;

dataSet.isDependent = defaultIsDependent;

dataSet.valuesArray.length = 6 \* 3 + 1;

return expect(matrixUtils.createMatrix(dataSet)).to.eventually.rejectedWith(expectedResult);

});

it('Should fail invalid array element.', function () {

expectedResult = 'Invalid type of value in array.';

dataSet.rows = defaultRows;

dataSet.columns = defaultColumns;

dataSet.valuesArray = [345, 65, 23, 168, 18, 18, 94, 0, 0, 187, 185, 98, 621, 87, 10, 255, 'a', 0];

dataSet.isDependent = defaultIsDependent;

return expect(matrixUtils.createMatrix(dataSet)).to.eventually.rejectedWith(expectedResult);

});

it('Should make successful comparison.', function () {

expectedResult = math.matrix([

[0.5664574696019713],

[0.06532925469423256],

[0.008718736194584409],

[0.15104864761034875]]);

//reuse existing dataSet

independentMatrix = math.matrix(

[[1, 345, 65, 23],

[1, 168, 18, 18],

[1, 94, 0, 0],

[1, 187, 185, 98],

[1, 621, 87, 10],

[1, 255, 0, 0]]);

//create dependent data set

dependentMatrix = math.matrix(

[[31.4],

[14.6],

[6.4],

[28.3],

[42.1],

[15.3]]);

return expect(Promise.resolve(matrixUtils.calculateCoefficients(independentMatrix, dependentMatrix))).to.eventually.become(expectedResult);

});

});