SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

PREDVIĐANJE USPJEHA UČENIKA

Vizualizacija podataka

Ivan Jakab

Osijek, 2020. godina

SADRŽAJ

1. UVOD

Koristeći skup podataka koji se nalazi uci repozitoriju, pod nazivom student performance, https://archive.ics.uci.edu/ml/datasets/student+performance, prikazuje se ovisnost uspješnosti učenika o parametrima sakupljenim u skupu podataka. Aplikacija se sastoji od dva dijela – servisnog, koji se bavi podacima i vizualnog.

Prikupljeni dataset prebačen je u JSON format, te se učitava u aplikaciji. Vizualni dio aplikacije prikuplja instrukcije od korisnika, prebacuje ih servisnom dijelu koji s podacima daje analitičke rezultate. Zatim vizualni dio vizualizira te podatke.

2. IZRADA PROJEKTA

Projekt je izrađen u javascript programskom jeziku, koristeći d3.js biblioteku. Za lakše stiliziranje, korišten je bootstrap. Ni jedna druga biblioteka nije korištena, no korištena je nova ES6 sintaksa.

Korišteni izvor podataka je u JSON formatu, odnosno lista objekata koji sadrže različite atribute:

```
const dataset = [{
    'sex': 'F',
    'age': 18,
    'famsize': 'GT3',
    'Medu': 4,
    'Fedu': 4,
    'Mjob': 'at_home',
    'Fjob': 'teacher',
    'traveltime': 2,
    'studytime': 2,
    'activities': 'no',
    'higher': 'yes',
    'freetime': 3,
    'goout': 4,
    'absences': 6,
    'G3': 6
}, {
    'sex': 'F',
    'age': 17,
    'famsize': 'GT3',
    'Medu': 1,
    'Fedu': 1,
    'Mjob': 'at_home',
    'Fjob': 'other',
    'traveltime': 1,
    'studytime': 2,
    'activities': 'no',
    'higher': 'yes',
    'freetime': 3,
    'goout': 3,
    'absences': 4,
    'G3': 6
}, {
    'sex': 'F',
    'age': 15,
    'famsize': 'LE3',
    'Medu': 1,
    'Fedu': 1,
    'Mjob': 'at_home',
    'Fjob': 'other',
    'traveltime': 1,
    'studytime': 2,
    'activities': 'no',
    'higher': 'yes',
    'freetime': 3,
    'goout': 2,
    'absences': 10,
    'G3': 10
}, {
```

```
    'sex': 'F',
    'age': 15,
    'famsize': 'GT3',
    'Medu': 4,
    'Fedu': 2,
    'Mjob': 'health',
    'Fjob': 'services',
    'traveltime': 1,
    'studytime': 3,
    'activities': 'yes',
    'higher': 'yes',
    'freetime': 2,
    'goout': 2,
    'absences': 2,
    'G3': 15
}, {
    'sex': 'F',
    'age': 16,
    'famsize': 'GT3',
    'Medu': 3,
    'Fedu': 3,
    'Mjob': 'other',
    'Fjob': 'other',
    'traveltime': 1,
    'studytime': 2,
    'activities': 'no',
    'higher': 'yes',
    'freetime': 3,
    'goout': 2,
    'absences': 4,
    'G3': 10
}]
```

Servisni dio aplikacije je klasa koja ima dvije glavne metode. Prva prima imena dvije varijable, te od njih izrađuje listu točaka za scater graf. Druga prima ime jedne varijable, te vrača raspodjelu te varijable u skupu podataka.

Osim ovih, postoji i metoda koja prima filtere, te cijeli proces može ograničiti na podskup koji zadovoljava neki uvijek. Kod je u nastavku.

```
const numericFields = {
    age: 'Broj godina',
    Medu: 'Edukacija majke',
    Fedu: 'Edukacija oca',
    traveltime: 'Vrijeme putovanja',
    studytime: 'Vrijeme učenja',
    freetime: 'Slobodno vrijeme',
    goout: 'Vrijeme izlazaka',
    absences: 'Izostanci',
    G3: 'Uspjeh',
}

const keywords = {
    sex: 'Spol',
    famsize: 'Veličina obitelji',
    Mjob: 'Posao majke',
    Fjob: 'Posao oca',
```

```
    activities: 'Izvannastavne aktivnosti',
    higher: 'Zeli nastaviti školovanje',
}

const allFields = {
    ...numericFields,
    ...keywords
}

class Analyzer {
    constructor(filterContainer) {
        this.filterContainer = filterContainer
        this.filters = [{
            field: 'G3',
            operator: 'greater than',
            value: 10
        }];
    }

    addFilter(filter) {
        this.filters.push(filter)
        this.getFilterHtml()
    }

    getFilterHtml() {
        let i = -1
        const htmlArr = this.filters.map(filter => {
            i++
            return `<span class="badge badge-danger" data-
index="${i}">${allFields[filter.field]} ${filter.operator} ${filter.value}
X</span>`
        })
        this.filterContainer.innerHTML = htmlArr.join(' ');
    }

    removeFilter(element) {
        const index = element.getAttribute('data-index')
        this.filters.splice(index, 1)
        this.getFilterHtml()
    }

    getFilteredSubset() {
        if(!this.filters.length) return dataset
        return dataset.filter(d => {
            for(const filter of this.filters) {
                const valueToCheck = d[filter.field]
                const targetValue = filter.value
                if(filter.operator === 'equals' && valueToCheck !=
targetValue) return false
                if(filter.operator === 'greater than' && valueToCheck <=
targetValue) return false
                if(filter.operator === 'less than' && valueToCheck >=
targetValue) return false
            }
            return true
        })
    }

    getNumericPoints(field1, field2) {
        const raw = this.getFilteredSubset().map(d => {
            return {
```

```
                x: d[field1],
                y: d[field2]
            }
        })
        // raw.sort(function(a, b){return a.x-b.x});
        return {
            x: raw.map(d => d.x),
            y: raw.map(d => d.y),
            xField: allFields[field1],
            yField: allFields[field2]
        }
    }

    getBarData(field) {
        const final = {}
        for (const row of this.getFilteredSubset()) {
            const value = row[field]
            if(final[value]) final[value] ++
            else final[value] = 1
        }
        return final
    }
}
```

Konačno, vizualni dio prima podatke iz podatkovnog (servisnog) dijela te na osnovu njih crta scatter i bar chart grafove. Pomoću biblioteke d3 prvo skalira podatke kako bi stali u 500x500 okvir, te manipulacijom svg elemenata crta grafove.

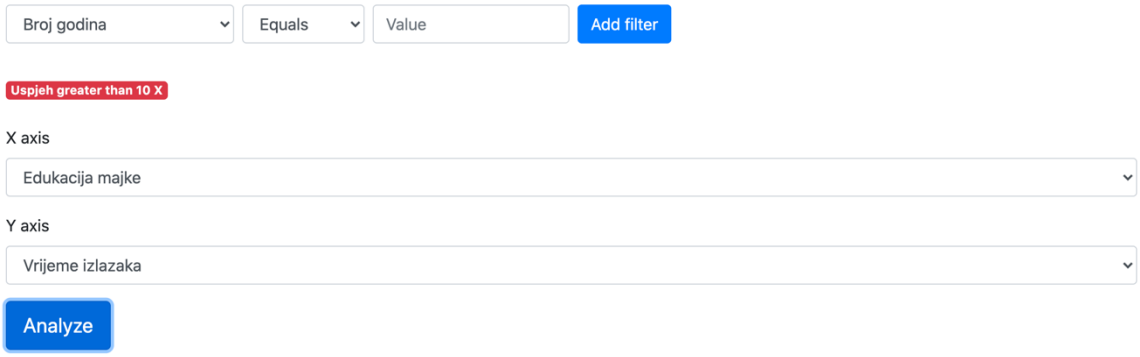
```
class Drawer {
    draw(data, barData) {
        const container = d3.select('#svgContainer')
        container.html('')

        this.drawScatter(data, container, data.xField, data.yField)
        this.drawBar(barData, container, data.xField)
    }

    drawBar(dataObj, container, xField) {
        const data = Object.values(dataObj)

        var margin = {top: 20, bottom: 70, left:40, right: 20};
        var width = 500 - margin.left - margin.right;
        var height = 500 - margin.top - margin.bottom;
        var barPadding = 4;
        var barWidth = width / data.length - barPadding;

        var x = d3.scale.ordinal()
            .domain(d3.range(data.length))
            .rangeRoundBands([0, width]);
        var y = d3.scale.linear()
            .domain([0, d3.max(data)])
            .range([height, 0]);

        container.append('br')
        container.append('br')
        var svg = container
```

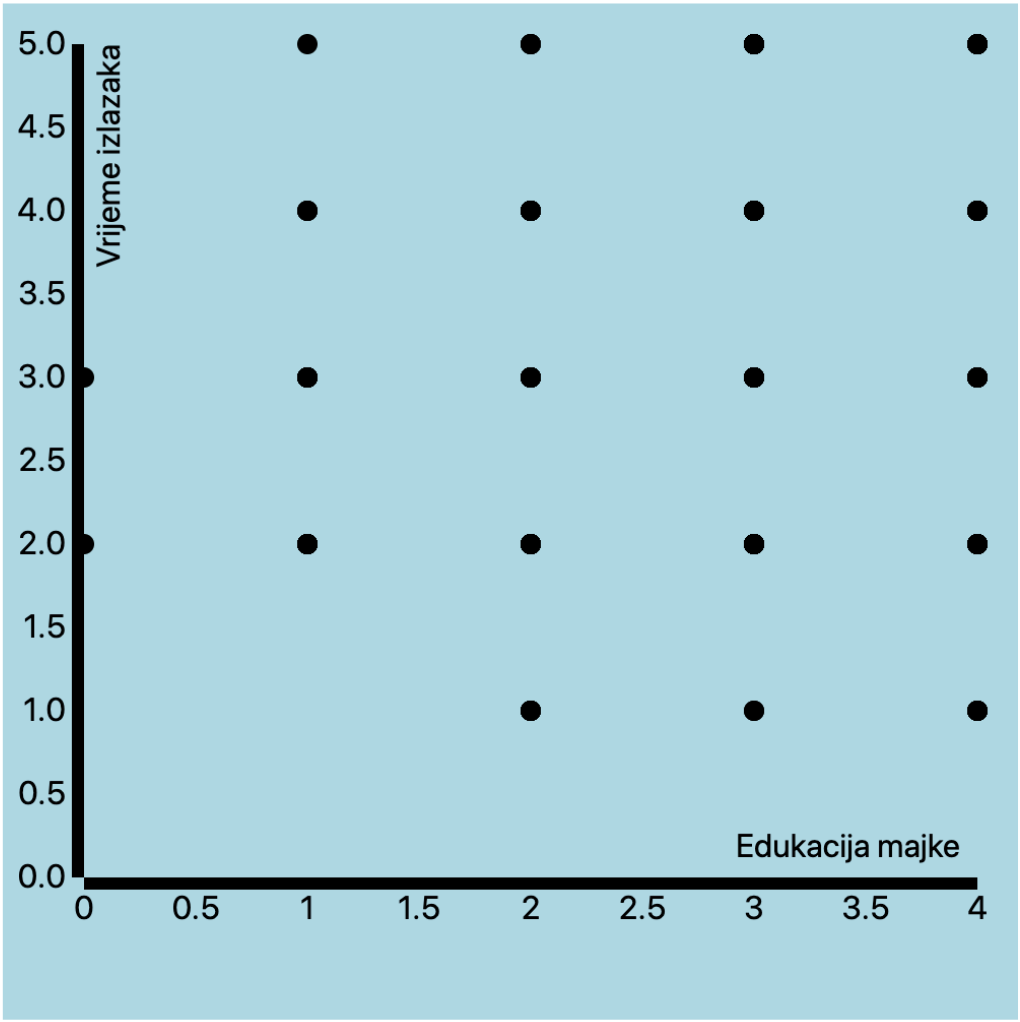
```
            .append("svg")
            .attr("width", width + margin.left + margin.right)
            .attr("height", height + margin.bottom + margin.top)
            .style("background-color", "lightblue")
            .append("g")
            .attr("transform", "translate(" + margin.left + "," +
margin.top + ")");

        var xAxis = d3.svg.axis()
            .scale(x)
            .orient("bottom")
            .tickFormat(function(d, i) { return ''; });
        var yAxis = d3.svg.axis()
            .scale(y)
            .orient("left")
            .ticks(10);
        svg.append("g")
            .attr("class", "y axis")
            .call(yAxis)

        var barchart = svg.selectAll("rect")
            .data(data)
            .enter()
            .append("rect")
            .attr("x", function(d, i) { return x(i); })
            .attr("y", y)
            .attr("height", function(d) { return height - y(d); })
            .attr("width", barWidth)
            .attr("fill", "blue");

        for(let i = 0; i < Object.keys(dataObj).length; i++) {
            svg.append("text")
                .attr("y", svgH-60)
                .attr('x', x(i) + 20)
                .text(Object.keys(dataObj)[i])
        }

        svg.append("g")
            .attr("class", "x axis")
            .attr("transform", "translate(0," + height + ")")
            .call(xAxis);
        svg.append("g")
            .attr("class", "y axis")
            .call(yAxis);
    }

    drawScatter(dataObj, container, xField, yField) {
        const dataY = dataObj.y
        const dataX = dataObj.x

        var margin = {top: 20, bottom: 70, left:40, right: 20};
        var width = 500 - margin.left - margin.right;
        var height = 500 - margin.top - margin.bottom;

        var x = d3.scale.linear()
            .domain([0, d3.max(dataX)])
            .range([0, width]);
        var y = d3.scale.linear()
            .domain([0, d3.max(dataY)])
            .range([height, 0]);
```

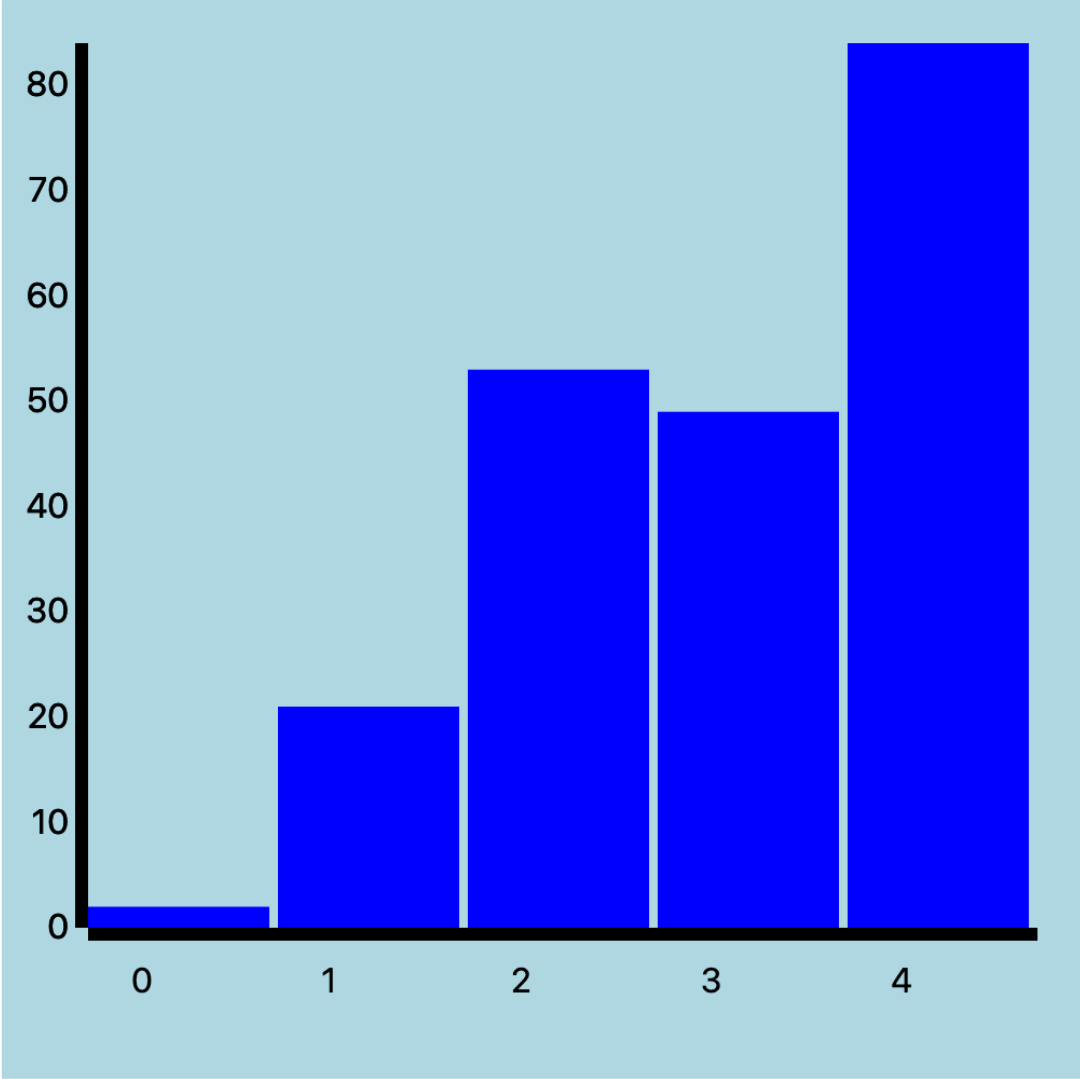
```
        var xAxis = d3.svg.axis()
            .scale(x)
            .orient("bottom")
            .tickFormat(d => {
                return d;
            });
        var yAxis = d3.svg.axis()
            .scale(y)
            .orient("left")
            .ticks(10);

        container.append('br')
        container.append('br')

        var svg = container
            .append("svg")
            .attr("width", width + margin.left + margin.right)
            .attr("height", height + margin.bottom + margin.top)
            .style("background-color", "lightblue")
            .append("g")
            .attr("transform", "translate(" + margin.left + "," +
margin.top + ")");
        const xAxisSvg = svg.append("g")
            .attr("class", "x axis")
            .attr("transform", "translate(0," + height + ")")
            .call(xAxis)
        xAxisSvg.selectAll("text")
            .style("text-anchor", "middle")
        xAxisSvg
            .append("text")
            .attr("x", 420)
            .attr("y", -10)
            .attr("dx", ".71em")
            .style("text-anchor", "end")
            .text(xField);
        svg.append("g")
            .attr("class", "y axis")
            .call(yAxis)
            .append("text")
            .attr("transform", "rotate(-90)")
            .attr("y", 6)
            .attr("dy", ".71em")
            .style("text-anchor", "end")
            .text(yField);

        for(let i = 0; i < dataX.length; i++) {
            svg.append('circle')
                .attr('cx', x(dataX[i]))
                .attr('cy', y(dataY[i]))
                .attr('r', 5)
                .attr('fill', 'black')
        }

    }
}

const drawer = new Drawer()
```

3. KONAČNA APLIKACIJA

Konačni izgled je nastavku. Numerički podaci su diskretni, pa je to odraženo i na scater graf.






80
70
60
50
40
30
20
10
0
0
1
2
3
4

4. NAPOMENE ZA KORIŠTENJE

Kompletan izvorni kod i povijest razvoja dostupni su na github repozitoriju na linku https://github.com/ijakab/student_performance_vizualization. Osim na githubu, sve je sadržano i u ovoj arhivi.

Kako bi se kod pokrenuo, potrebno je na bilo koji način poslužiti aplikaciju, odnosno statični direktorij. Glavna datoteka je index.html. U projektu je već dodana datoteka server.js, koja se može pokrenuti s node interpreterom (node server.js), i ta će komanda slušati upite na portu 8080.

Ako node interpreter nije dostpan, bilo koja druga metoda može poslužiti statički direktorij (php i apache, pyhton server etc.) Uz to, dodan je i Dockerfile, kojega je moguće buildati i pokrenuti. (docker build ., dobije se id docker imagea, zatim docker run id).

Kod koristi novu ES6 sintaksu, pa neće raditi u starijim browserima. Preporučeno koristiti google chrome.

5. LITERATURA

Twitter bootstrap

https://getbootstrap.com/docs/4.5/getting-started/introduction/

d3.js

https://d3js.org/

Student performance dataset

https://archive.ics.uci.edu/ml/datasets/student+performance