SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Projektni zadatak iz predmeta
VIZUALIZACIJA PODATAKA

Top 250 IMDB

Student: Adrijan Malinović, DRD 1.god

Mentor: Josip Jop, Časlav Livada

U Osijeku, 2024

SADRŽAJ

1. KV1 - Definiranje projektnog zadatka

1.1. Projektni zadatak

Naziv zadatka: Vizualizacija top 250 najbolje ocijenjenih filmova sa IMBD-a 2023. godine

Opis problema: Podatci sadrže informacije o 250 najpopularnijih/najboljih filmova 2023. godine. Potrebno je proučiti podatke radi stvaranja veza I zaključaka vezano uz razne aspekte ovih filmova kao što su: godinja izdanja, trajanje, ocjena , sveukupni rank, žanra itd.

Opis zadatka: Vizualizirati podatke i sazanja o najbolje ocijenjenim filmovima u 2023. godini.

Cilj projekta: Steći uvid u odnose među podatcima koji opisiju filmove iz ovog podatkovnog skupa.

Poveznica na git repozitorij projekta: https://github.com/adrijanmalinovic/Vizualizacija-Podataka-KV

1.2. Podatci

Z-1.2.1. *U ovom zadatku potrebno je pronaći odgovarajuće izvore podataka koji će se koristiti za rješavanje problema definiranog u prvom zadatku. Važno je osigurati da su podaci kvalitetni i relevantni za problem koji se rješava te da su dostupni za upotrebu.*

https://www.kaggle.com/datasets/ashishjangra27/imdb-top-250-movies

Z-1.2.2. *Potrebno je opisati odabrane podatke*

Podatkovni skup sadrži podatke o 250 filmova prikupljenih 2023. g. Podatci su u csv formatu, a sadrže razne stupce kao što su ime filma, rang po ocjenjenosti (1 –“najbolji”, 250 – “najlošij”), naslovna slika, godina izdanja, glasovi na IMBD-u, ocjenu sa IMBD-a, trajanj, listu direktora, glumaca, pisaca te druge podatke.

1.3. Obrada podataka

Z-1.3.1. *Obraditi prikupljene podatke i povezati ih kako bi se stvorio cjelovit skup podataka. Ovo uključuje čišćenje i obradu podataka, kao i provjeru njihove konzistentnosti, aktualnosti, cjelovitosti, tj. kvalitete i ispravnosti.*

Podatke treba restrukturirati u JSON format radi lakšeg korištenja u D3 biblioteci. Također je potrebno obrisati razne stupce sa metapodatcima vezanim uz dataset kao što su ID filma, glumaca I direktora. Oni se neće koristiti u vizualizaciji. Neke stupci kao što su direktori i glumci, koji su originalno obliku jednog stringa, je poželjno prebaciti u polja stringova radi lakšeg pristupa određenim elementima ako za

to bude potrebe. Također je poželjno preimenovati imena JSON ključeva radi konzistentnosti i jasnoće. Budući da u podatkovnom skupu nema nedostajućih ili pogrešnih vrijednosti, u tom pogledo nije potrebno ništa mijenjati.


```
[
    {
        "rank": 7,
        "year": 2003,
        "duration": 201,
        "title": "The Lord of the Rings: The Return of the King",
        "votes": 1786498,
        "rating": 9,
        "image": "https://m.media-amazon.com/images/M/MV5BOGM4ZTE3MmUtY2EzYy00YTYyLTkwZTUtYWFhNTBiNGY1ZDg0XkEyXkFqcGdeQXVyMzExODEzNDA@._V1_QL75_UX380_CR0",
        "genres": [
            "Action",
            "Adventure",
            "Drama"
        ],
        "actors": [
            "Elijah Wood",
            "Viggo Mortensen",
            "Ian McKellen",
            "Orlando Bloom",
            "Noel Appleby",
            "Ali Astin",
            "Sean Astin",
            "David Aston",
            "John Bach",
            "Sean Bean",
            "Cate Blanchett",
            "Billy Boyd",
            "Sadwyn Brophy",
            "Alistair Browning",
            "Marton Csokas",
            "Richard Edge",
            "Jason Fitch",
            "Bernard Hill"
        ],
        "directors": [
            "Peter Jackson"
        ],
        "writers": [
            "J.R.R. Tolkien",
            "Fran Walsh",
            "Philippa Boyens"
        ]
    }
]
```

1.4. Relevantne vrste prikaza za korištene podatke

Z-1.4.1. *Predložiti moguće načine prikaza podataka koji će pomoći u razumijevanju podataka i rješavanju problema koji je postavljen u prvom zadatku. Ovo može uključivati odabir najprikladnijeg načina vizualizacije podataka, ali to je zadatak iduće vježbe.*

Koristit će se ponajviše stupčasti dijagrami, histogrami, dijagrami raspršenja, liste i text, te *grid* prikazi sa slikama. Ideja je vidjeti, primjerice, kako na rangiranje/ocjenu filma ovisi trajanje filma, koji žanrovi su najbolje rangirani, utječe li zanimljivija naslovna slika na ocjenu filma itd.

2. KV2 - Dizajn vizualizacije podataka.

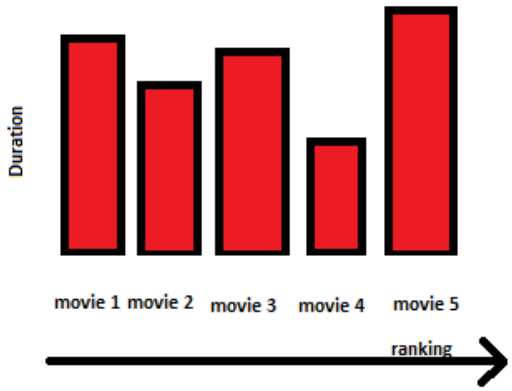
2.1. Pitanja na koja vizualizacija daje odgovor

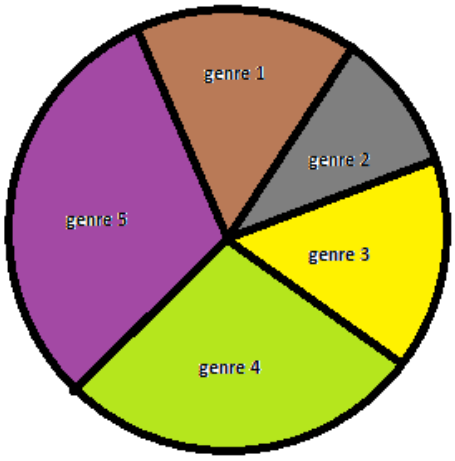
Z-2.1.1. Popis pitanja na koja vizualizacija daje odgovor.

Koji žanrovi filmova su najbolje ocjenjeni/gledani? Jesu li bolje ocjenjeni klasici (stariji) ili moderni filmovi? Utječe li trajanje filma na ocjenu? Imaju li bolje ocjenjeni filmovi zanimljivije slike? Kojih godina su filmovi iz podataka izlazili i koliko njih?

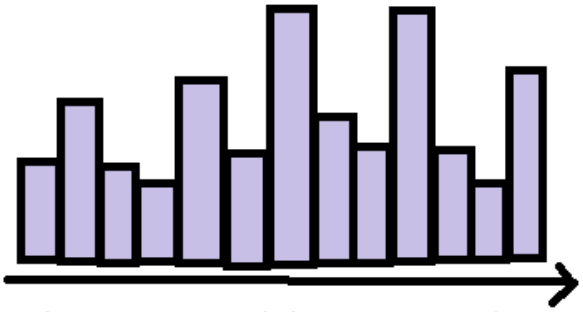
2.2. Skica vizualizacije podataka

Z-2.2.1. *Izraditi skice konačne vizualizacije podataka, koja će uključivati sve elemente potrebne za rješavanje problema. Ovo uključuje različite tipove grafikona, dijagrama i drugih vizualnih elemenata koji će biti uključeni u vizualizaciju podataka.*





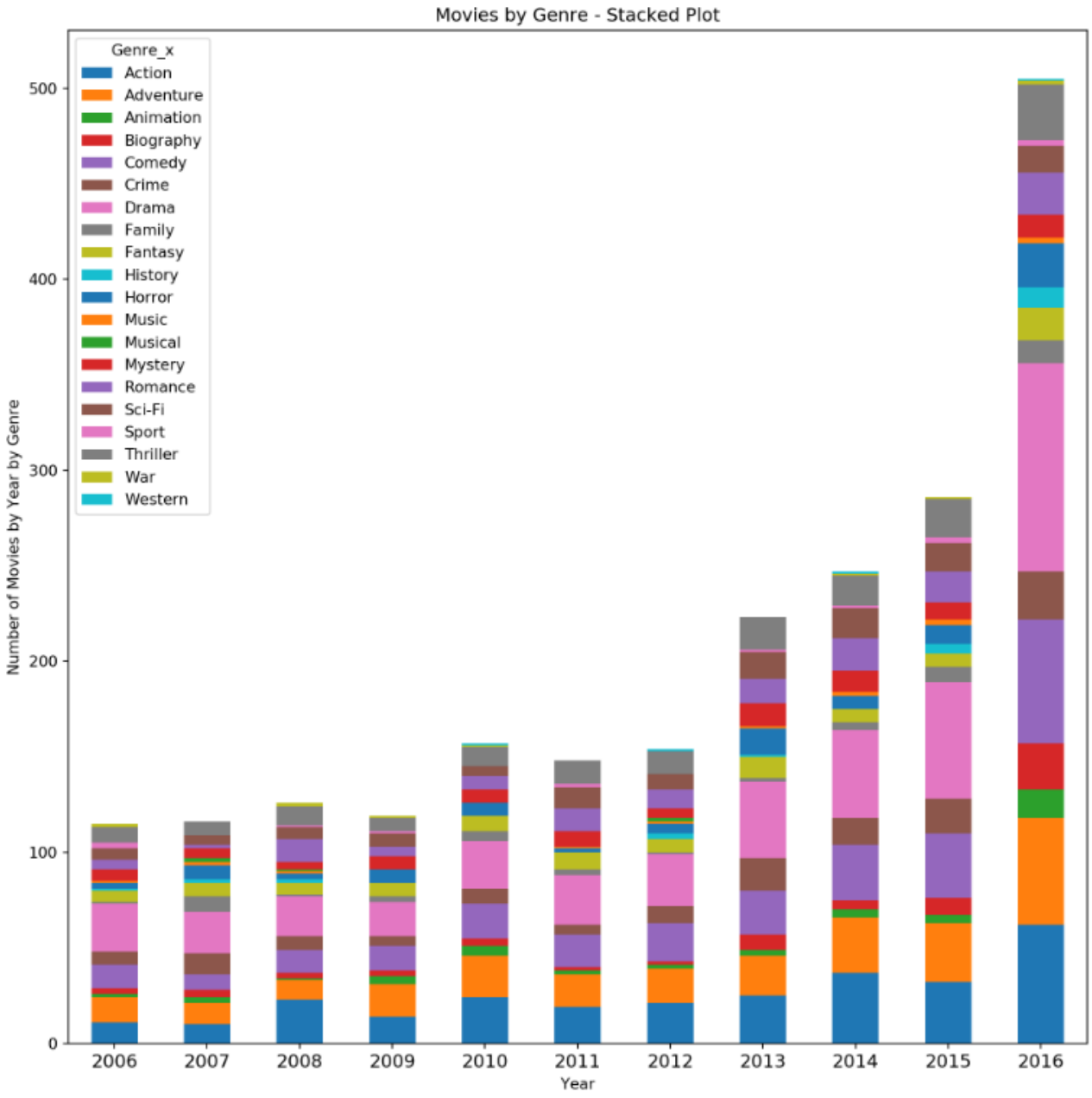
Percentages of specific genres in the whole movie set

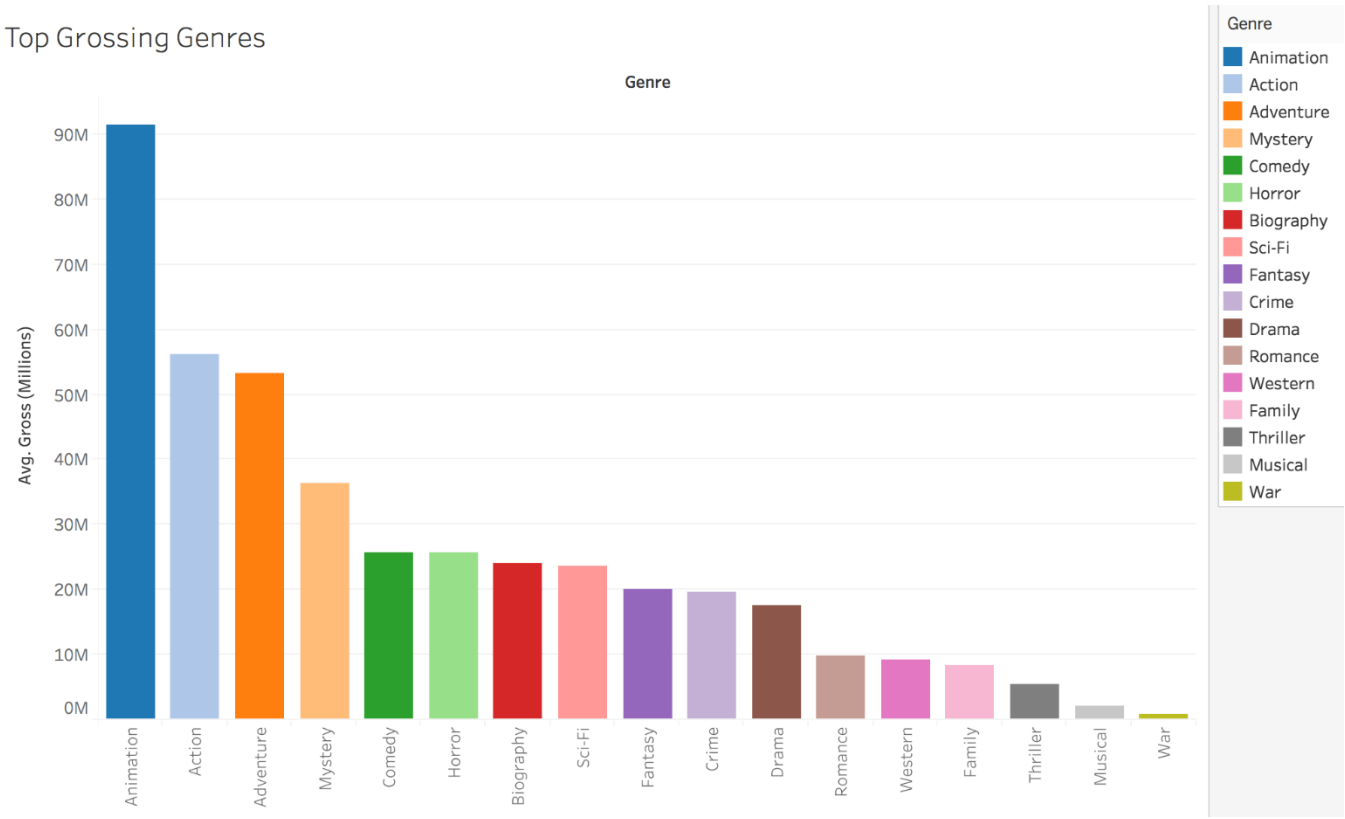


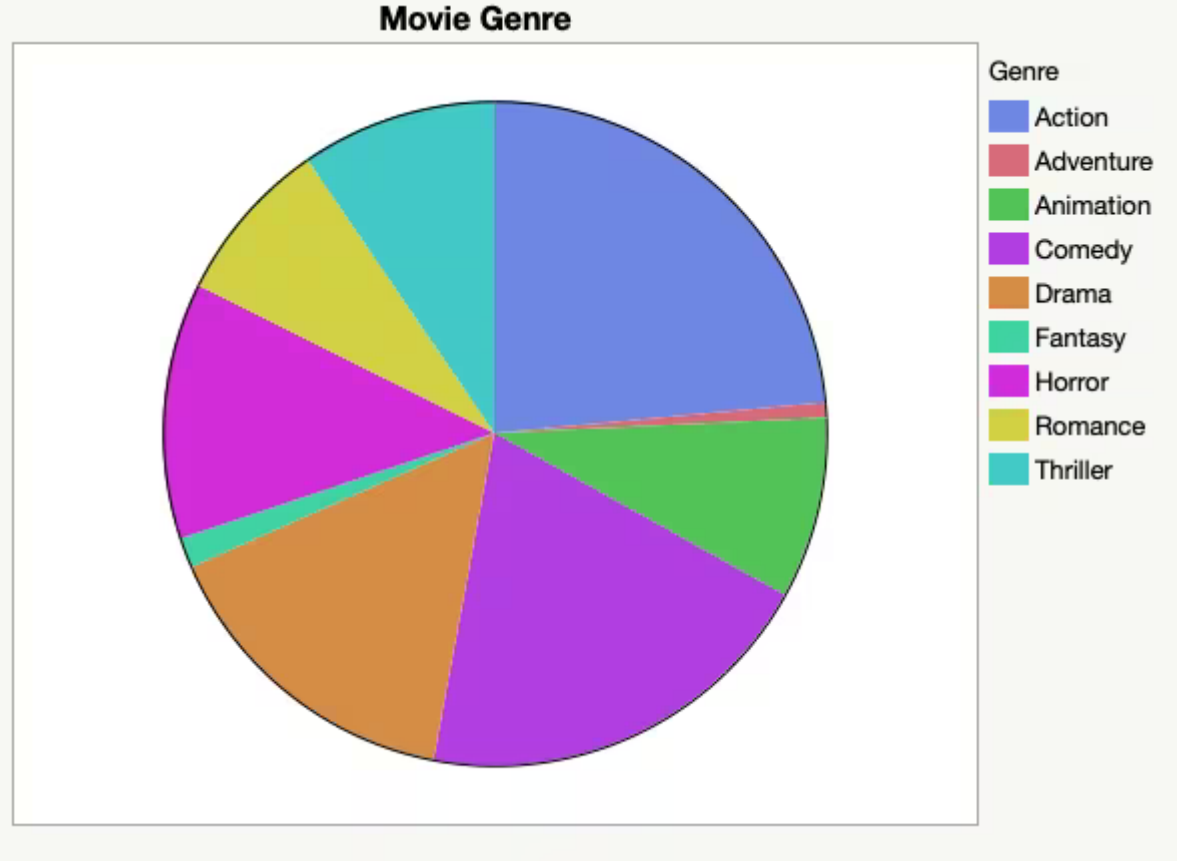
Histogram of years the movies in the set were released in

2.3. Postojeća rješenja i primjeri

Z-2.3.1. *Pretražiti dostupne stranice sa zbirkama vizualizacija podataka koje su korisne u ovom projektu.*


Movies by Genre - Stacked Plot
Genre_x
Action
Adventure
Animation
Biography
Comedy
Crime
Drama
Family
Fantasy
History
Horror
Music
Musical
Mystery
Romance
Sci-Fi
Sport
Thriller
War
Western
Number of Movies by Year by Genre
0
100
200
300
400
500
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
Year





2.4. Prilagodba podataka

Z-2.4.1. Pripremiti podatke za vizualizaciju.

Podatci su pripremljeni i očišćeni u KV1

Z-2.4.2. *Odabrati odgovarajući oblik (engl. format) podataka.*

Podatci su spremljeni u JSON obliku

Z-2.4.3. *Urediti podatke za vizualizaciju i prikazati ih u tablici ili drugom prikladnom obliku.*

Podatci su već u tabličnom formatu I prikladni su za vizualizaciju

Z-2.4.4. *Pokazati slikom da su podatci uspješno prilagođeni i prikazani na grafičkom prikazu.*


```
{
    "rank": 236,
    "year": 2003,
    "duration": 143,
    "title": "Pirates of the Caribbean: The Curse of the Black Pearl",
    "votes": 1099699,
    "rating": 8.1,
    "image": "https://m.media-amazon.com/images/M/MV5BZjUxNmEwOGItMTBmYi00MWQ1LWExY2MtNDUxMjI0OWM4M2NiXkEyXkFqcGdeQXVyMjUzOTY1NTc@._V1_QL75_UX380_CR0",
    "genres": [
        "Action",
        "Adventure",
        "Fantasy"
    ],
    "actors": [
        "Johnny Depp",
        "Geoffrey Rush",
        "Orlando Bloom",
        "Keira Knightley",
        "Jack Davenport",
        "Jonathan Pryce",
        "Lee Arenberg",
        "Mackenzie Crook",
        "Damian O'Hare",
        "Giles New",
        "Angus Barnett",
        "David Bailie",
        "Michael Berry Jr.",
        "Isaac C. Singleton Jr.",
        "Kevin McNally",
        "Treva Etienne",
        "Zoe Saldana",
        "Guy Siner"
    ],
    "directors": [
        "Gore Verbinski"
    ],
    "writers": [
        "Ted Elliott",
        "Terry Rossio",
        "Stuart Beattie"
    ]
},
```


2.5. Boje i podatci

Z-2.5.1. Popis korištenih boja s pripadajućim obrazloženjem.

U ovom projektu boje neće biti od velike važnosti, eventualno će se koristiti za pomoć u već postojećim dijagramima I prikazima. Primjerice, u histogramu će veće vrijednosti imate više udjela neke boje da se vidi dodatno razlika između vrijednosti.

3. KV3 - Izrada prototipne vizualizacije podataka

3.1. Osnovne funkcionalnosti i ponašanja

Projekt će imati jednu početnu stranicu za opće informacije kao što su github repozitorij i slično. Svaka podstranica će imati svoje vizualizacije, a do tih podstranica moći će se navigirati sa izbornika na lijevoj strani. Koristit će se slike, histogrami te dijagrami raspršenja

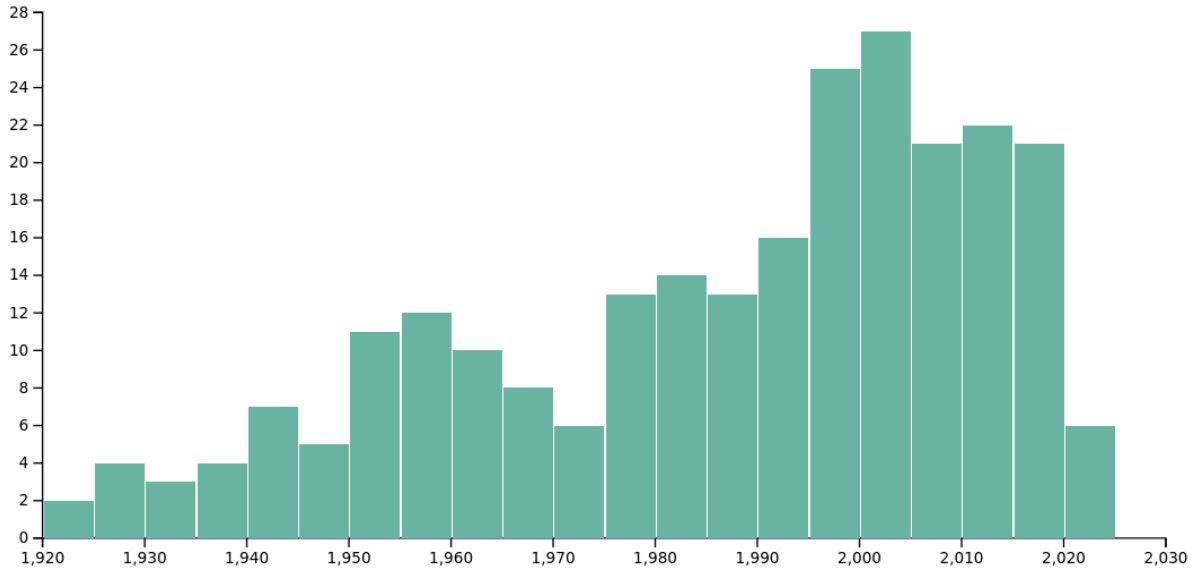
3.2. Napredne funkcionalnosti i ponašanja:

Korisnik će na svakoj vizualizaciji imati više padajućih izbornika za kontrolu vizualizacije ovisno o tome što korisnik hoće prikazati i kakve informacije/ovisnosti podataka ga zanimaju, a osim toga moći će se mišem prelaziti preko određenih dijelova vizualizacije za više informacija ili za informacije koje možda nisu očito vidljive.

3.3. Implementacija funkcionalnosti i ponašanja

```
const svg = d3.select("#chart")
    .append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
    .attr("transform", `translate(${margin.left},${margin.top})`);

// Set up scales and axis
const x = d3.scaleLinear()
    .domain(d3.extent(values))
    .nice()
    .range([0, width]);

const histogram = d3.histogram()
    .value(d => d)
    .domain(x.domain())
    .thresholds(x.ticks(20));

const bins = histogram(values);

const y = d3.scaleLinear()
    .domain([0, d3.max(bins, d => d.length)])
    .nice()
    .range([height, 0]);

svg.append("g")
    .call(d3.axisLeft(y));

svg.append("g")
    .attr("transform", `translate(0,${height})`)
    .call(d3.axisBottom(x));
```

3.4. Prikaz osnovne funkcionalnosti i ponašanja



4. KV4 - Izrada konačne vizualizacije podataka

4.1. Implementacija funkcionalnosti i ponašanja

```html
<!-- collection.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Movie Collection</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <div class="sidebar-container">
        <!-- Sidebar will be imported here -->
    </div>

    <div class="content">
        <div class="container">
            <h1>Movie Collection</h1>
            <div class="controls">
                <select id="sort-by" class="dropdown">
                    <option value="rank-asc">Rank (Ascending)</option>
                    <option value="rank-desc">Rank (Descending)</option>
                    <option value="year-asc">Year (Ascending)</option>
                    <option value="year-desc">Year (Descending)</option>
                    <option value="duration-asc">Duration (Ascending)</option>
                    <option value="duration-desc">Duration (Descending)</option>
                    <option value="votes-asc">Votes (Ascending)</option>
                    <option value="votes-desc">Votes (Descending)</option>
                    <option value="rating-asc">Rating (Ascending)</option>
                    <option value="rating-desc">Rating (Descending)</option>
                </select>
                <select id="filter-genre" class="dropdown">
                    <option value="">All Genres</option>
                </select>
            </div>
            <div id="movie-grid" class="grid"></div>
        </div>
    </div>



    <script src="https://d3js.org/d3.v6.min.js"></script>
    <script src="collection.js"></script>
    <script>
        // JavaScript to import sidebar.html into sidebar-container div
        fetch('sidebar.html')
            .then(response => response.text())
            .then(html => {
                document.querySelector('.sidebar-container').innerHTML = html;
            });
    </script>
</body>
</html>
```

```
/* collection.js */
d3.json("movies.json").then(data => {
    console.log("Data loaded:", data);

    let movies = data.map(movie => {
        movie.rank = +movie.rank;
        movie.duration = +movie.duration;
        movie.votes = +movie.votes;
        movie.rating = +movie.rating;
        movie.year = +movie.year; // Parse year as a number
        return movie;
    });

    // Check if movies data is correctly processed
    console.log("Processed movies:", movies);

    // Get all unique genres
    let genres = new Set();
    movies.forEach(movie => {
        movie.genres.forEach(genre => genres.add(genre));
    });

    // Populate genre filter
    const genreFilter = d3.select("#filter-genre");
    genres.forEach(genre => {
        genreFilter.append("option").attr("value", genre).text(genre);
    });

    // Initial rendering
    renderGrid(movies);

    // Event listeners for sorting and filtering
    d3.select("#sort-by").on("change", () => {
        const sortedMovies = sortMovies(filterMovies(movies));
        console.log("Sorted movies:", sortedMovies);
        renderGrid(sortedMovies);
    });

    d3.select("#filter-genre").on("change", () => {
        const filteredMovies = filterMovies(movies);
        console.log("Filtered movies:", filteredMovies);
        renderGrid(filteredMovies);
    });

    // Sorting function
    function sortMovies(movies) {
        const sortBy = d3.select("#sort-by").node().value;
        const [key, order] = sortBy.split("-");
        return movies.slice().sort((a, b) => {
            if (order === "asc") {
                return a[key] - b[key];
            } else {
                return b[key] - a[key];
            }
        });
    }

    // Filtering function
    function filterMovies(movies) {
        const selectedGenre = d3.select("#filter-genre").node().value;
        return selectedGenre === "" ? movies : movies.filter(movie => movie.genres.includes(selectedGenre));
    }

    // Render grid function
    function renderGrid(movies) {
        console.log("Rendering grid with movies:", movies);
        const grid = d3.select("#movie-grid").html("");
        movies.forEach(movie => {
            const movieDiv = grid.append("div").attr("class", "movie");
            movieDiv.append("img").attr("src", movie.image).attr("alt", movie.title);
            movieDiv.append("div").attr("class", "movie-title").text(movie.title);
            const movieInfo = movieDiv.append("div").attr("class", "movie-info");
            movieInfo.html(`
                <div style="display: flex; gap: 20px;">
                    <div>
                        <p><strong>Title:</strong><br>${movie.title}</p>
                        <p><strong>Rank:</strong><br>${movie.rank}</p>
                        <p><strong>Rating:</strong><br>${movie.rating}</p>
                        <p><strong>Votes:</strong><br>${movie.votes}</p>
                        <p><strong>Duration:</strong><br>${movie.duration} minutes</p>
                        <p><strong>Year:</strong><br>${movie.year}</p>
                        <p><strong>Genres:</strong><br>${movie.genres.join(', ')}</p>
                    </div>
                    <div>
                        <p><strong>Directors:</strong><br><em>${movie.directors.join(', ')}</em></p>
                        <p><strong>Writers:</strong><br><em>${movie.writers.join(', ')}</em></p>
                        <p><strong>Actors:</strong><br><em>${movie.actors.slice(0, 6).join(', ')}${movie.actors.length > 6 ? '...' : ''}</em></p>
                    </div>
                </div>
            `);
        });
    }
}).catch(error => {
    console.error('Error loading the data:', error);
});
```

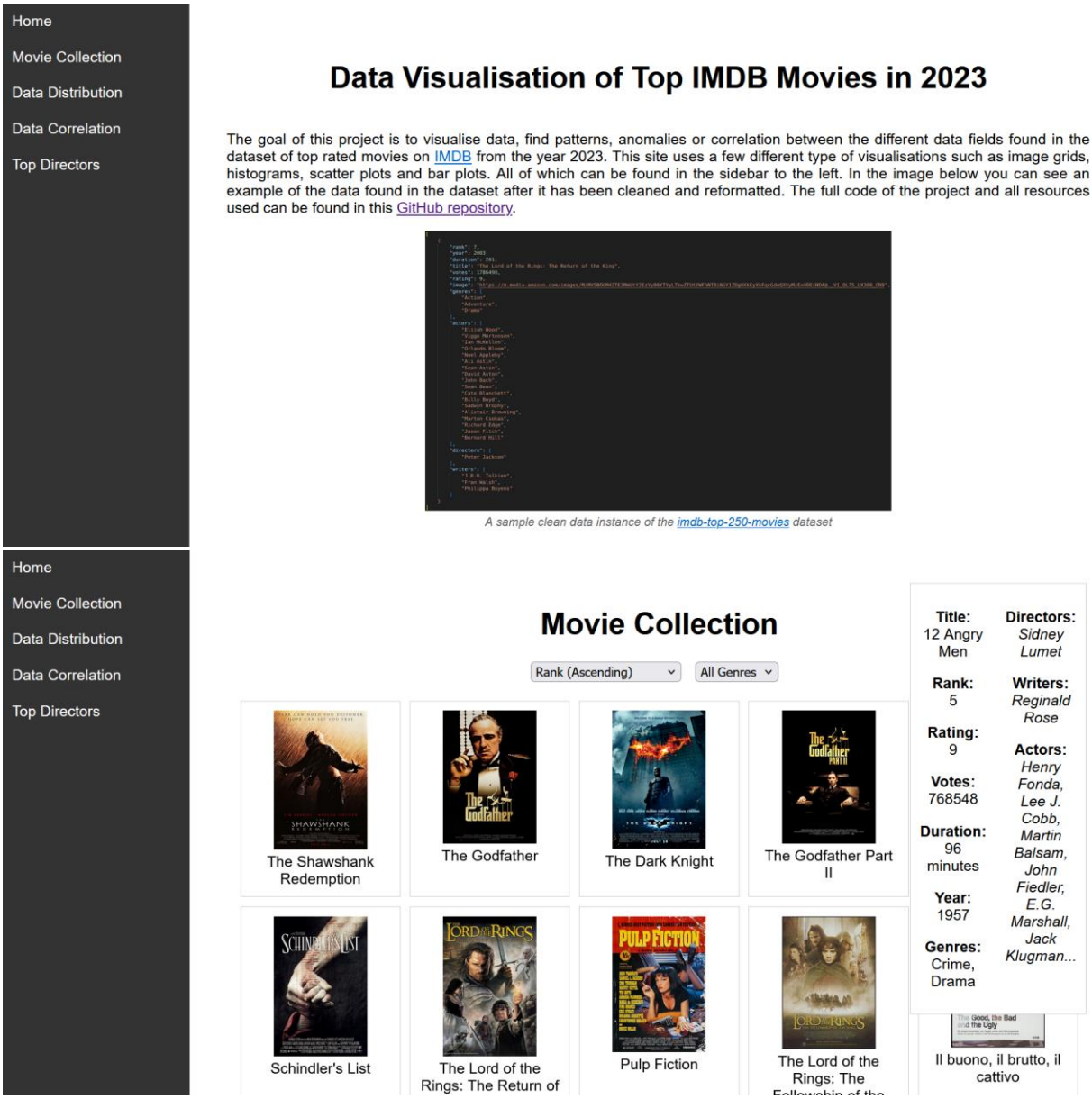
```html
<!-- distribution.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Data Distribution</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <div class="sidebar-container">
        <!-- Sidebar will be imported here -->
    </div>

    <div class="content">
        <div class="container">
            <h1>Data Distribution</h1>
            <div class="controls">
                <label for="metric">Data:</label>
                <select id="metric" class="dropdown">
                    <option value="year">Year</option>
                    <option value="duration">Duration</option>
                    <option value="votes">Votes</option>
                    <option value="rating">Rating</option>
                </select>
                <label for="genre">Genre:</label>
                <select id="genre" class="dropdown">
                    <option value="">All Genres</option>
                </select>
            </div>
            <div id="chart"></div>
        </div>
    </div>

    <script src="https://d3js.org/d3.v6.min.js"></script>
    <script src="distribution.js"></script>
    <script>
        // JavaScript to import sidebar.html into sidebar-container div
        fetch('sidebar.html')
            .then(response => response.text())
            .then(html => {
                document.querySelector('.sidebar-container').innerHTML = html;
            });
    </script>
</body>
</html>
```

```
// distribution.js */
d3.json("movies.json").then(data => {
    console.log("Data loaded:", data);

    let movies = data.map(movie => {
        movie.rank = +movie.rank;
        movie.duration = +movie.duration;
        movie.votes = +movie.votes;
        movie.rating = +movie.rating;
        movie.year = +movie.year;
        return movie;
    });

    // Set all unique genres
    let genres = new Set();
    movies.forEach(movie => {
        movie.genres.forEach(genre => genres.add(genre));
    });

    // Populate genre filter
    const genreFilter = d3.select("#genre");
    genres.forEach(genre => {
        genreFilter.append("option").attr("value", genre).text(genre);
    });

    // Initial rendering
    renderChart(movies, "year", "");

    // Event listeners for metric and genre selection
    d3.select("#metric").on("change", () => {
        const metric = d3.select("#metric").node().value;
        const genre = d3.select("#genre").node().value;
        renderChart(movies, metric, genre);
    });

    d3.select("#genre").on("change", () => {
        const metric = d3.select("#metric").node().value;
        const genre = d3.select("#genre").node().value;
        renderChart(movies, metric, genre);
    });

    // Render chart function
    function renderChart(movies, metric, genre) {
        console.log(`Rendering chart with metric: ${metric}, genre: ${genre}`);
        let filteredMovies = movies;
        if (genre) {
            filteredMovies = movies.filter(movie => movie.genres.includes(genre));
        }

        const values = filteredMovies.map(movie => movie[metric]);

        // Set up chart dimensions
        const margin = { top: 20, right: 30, bottom: 60, left: 60 },
            width = 800 - margin.left - margin.right,
            height = 400 - margin.top - margin.bottom;

        // Remove any existing chart
        d3.select("#chart").html("");

        const svg = d3.select("#chart")
            .append("svg")
            .attr("width", width + margin.left + margin.right)
            .attr("height", height + margin.top + margin.bottom)
            .append("g")
            .attr("transform", `translate(${margin.left},${margin.top})`);

        // Set up scales and axis
        const x = d3.scaleLinear()
            .domain(d3.extent(values))
            .nice()
            .range([0, width]);

        const histogram = d3.histogram()
            .value(d => d)
            .domain(x.domain())
            .thresholds(x.ticks(20));

        const bins = histogram(values);

        const y = d3.scaleLinear()
            .domain([0, d3.max(bins, d => d.length)])
            .nice()
            .range([height, 0]);

        svg.append("g")
            .call(d3.axisLeft(y));

        svg.append("g")
            .attr("transform", `translate(0,${height})`)
            .call(d3.axisBottom(x).ticks(0).tickSizeOuter(0));

        // Add x axis label
        svg.append("text")
            .attr("class", "x-label")
            .attr("text-anchor", "middle")
            .attr("x", width / 2)
            .attr("y", height + margin.bottom - 10)
            .text(metric.charAt(0).toUpperCase() + metric.slice(1));

        // Add Y axis label
        svg.append("text")
            .attr("class", "y-label")
            .attr("text-anchor", "middle")
            .attr("transform", "rotate(-90)")
            .attr("y", -margin.left + 20)
            .attr("x", -height / 2)
            .text("Count");

        // Tooltip setup
        const tooltip = d3.select("body").append("div")
            .attr("class", "tooltip")
            .style("position", "absolute")
            .style("visibility", "hidden")
            .style("background-color", "white")
            .style("border", "solid 1px #ccc")
            .style("padding", "10px")
            .style("border-radius", "5px");

        // Draw bars
        svg.selectAll("rect")
            .data(bins)
            .enter().append("rect")
            .attr("x", 1)
            .attr("transform", d => `translate(${x(d.x0)},${y(d.length)})`)
            .attr("width", d => x(d.x1) - x(d.x0) - 1)
            .attr("height", d => height - y(d.length))
            .style("fill", "#69b3a2")
            .on("mouseover", (event, d) => {
                const metricLabel = metric.charAt(0).toUpperCase() + metric.slice(1);
                tooltip.style("visibility", "visible")
                    .html(`<strong>Count:</strong><br>${d.length}<br><br><strong>${metricLabel}:</strong><br>${d.x0} - ${d.x1}`);
            })
            .on("mousemove", event => {
                tooltip.style("top", (event.pageY - 10) + "px")
                    .style("left", (event.pageX + 10) + "px");
            })
            .on("mouseout", () => {
                tooltip.style("visibility", "hidden");
            });
    }
}).catch(error => {
    console.error('Error loading the data:', error);
});
```

```html
<!-- correlation.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Data Correlation</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <div class="sidebar-container">
        <!-- Sidebar will be imported here -->
    </div>

    <div class="content">
        <div class="container">
            <h1>Data Correlation</h1>
            <div class="controls">
                <label for="x-metric">X-axis:</label>
                <select id="x-metric" class="dropdown">
                    <option value="year">Year</option>
                    <option value="duration">Duration</option>
                    <option value="votes">Votes</option>
                    <option value="rating">Rating</option>
                </select>
                <label for="y-metric">Y-axis:</label>
                <select id="y-metric" class="dropdown">
                    <option value="rating">Rating</option>
                    <option value="votes">Votes</option>
                    <option value="duration">Duration</option>
                    <option value="year">Year</option>
                </select>
            </div>
            <div id="chart"></div>
        </div>
    </div>

    <script src="https://d3js.org/d3.v6.min.js"></script>
    <script src="correlation.js"></script>
    <script>
        // JavaScript to import sidebar.html into sidebar-container div
        fetch('sidebar.html')
            .then(response => response.text())
            .then(html => {
                document.querySelector('.sidebar-container').innerHTML = html;
            });
    </script>
</body>
</html>
```

```
/* correlation.js */
d3.json("movies.json").then(data => {
    console.log("Data loaded:", data);
    let movies = data.map(movie => {
        movie.rank = +movie.rank;
        movie.duration = +movie.duration;
        movie.votes = +movie.votes;
        movie.rating = +movie.rating;
        movie.year = +movie.year;
        return movie;
    });

    // Initial rendering
    renderChart(movies, "year", "rating");

    // Event listeners for metric selection
    d3.select("#x-metric").on("change", () => {
        const xMetric = d3.select("#x-metric").node().value;
        const yMetric = d3.select("#y-metric").node().value;
        renderChart(movies, xMetric, yMetric);
    });

    d3.select("#y-metric").on("change", () => {
        const xMetric = d3.select("#x-metric").node().value;
        const yMetric = d3.select("#y-metric").node().value;
        renderChart(movies, xMetric, yMetric);
    });

    // Render chart function
    function renderChart(movies, xMetric, yMetric) {
        console.log(`Rendering chart with x-axis metric: ${xMetric}, y-axis metric: ${yMetric}`);
        const margin = { top: 20, right: 30, bottom: 40, left: 90 },
            width = 800 - margin.left - margin.right,
            height = 600 - margin.top - margin.bottom;

        // Remove any existing chart
        d3.select("#chart").html("");

        const svg = d3.select("#chart")
            .append("svg")
            .attr("width", width + margin.left + margin.right)
            .attr("height", height + margin.top + margin.bottom)
            .append("g")
            .attr("transform", `translate(${margin.left},${margin.top})`);

        // Set up scales and axis
        const xScale = d3.scaleLinear()
            .domain([d3.extent(movies, d => d[xMetric])])
            .range([0, width])
            .nice();

        const yScale = d3.scaleLinear()
            .domain([d3.extent(movies, d => d[yMetric])])
            .range([height, 0])
            .nice();

        const xAxis = d3.axisBottom(xScale).ticks(8).tickSizeOuter(0);
        const yAxis = d3.axisLeft(yScale);

        svg.append("g")
            .attr("transform", `translate(0,${height})`)
            .call(xAxis);

        svg.append("g")
            .call(yAxis);

        // Add x axis label
        svg.append("text")
            .attr("class", "x-label")
            .attr("text-anchor", "middle")
            .attr("x", width / 2)
            .attr("y", height + margin.bottom - 10)
            .text(xMetric.charAt(0).toUpperCase() + xMetric.slice(1));

        // Add y axis label
        svg.append("text")
            .attr("class", "y-label")
            .attr("text-anchor", "middle")
            .attr("transform", "rotate(-90)")
            .attr("y", -margin.left + 20)
            .attr("x", -height / 2)
            .text(yMetric.charAt(0).toUpperCase() + yMetric.slice(1));

        // Tooltip setup
        const tooltip = d3.select("body").append("div")
            .attr("class", "tooltip")
            .style("position", "absolute")
            .style("visibility", "hidden")
            .style("background-color", "white")
            .style("border", "solid 1px #ccc")
            .style("padding", "10px")
            .style("border-radius", "5px");

        // Create a map to store movies at each position
        const positionMap = new Map();

        movies.forEach(movie => {
            const key = `${xScale(movie[xMetric])},${yScale(movie[yMetric])}`;
            if (!positionMap.has(key)) {
                positionMap.set(key, []);
            }
            positionMap.get(key).push(movie);
        });

        // Draw scatter plot points
        svg.selectAll(".dot")
            .data(Array.from(positionMap.entries()))
            .enter().append("circle")
            .attr("class", "dot")
            .attr("cx", d => d[0].split(',')[0])
            .attr("cy", d => d[0].split(',')[1])
            .attr("r", d => Math.min(5, 2 + d[1].length)) // Increase size for overlapping points
            .style("fill", "red")
            .on("mouseover", (event, d) => {
                const moviesAtPoint = d[1];
                let tooltipContent = moviesAtPoint.map(movie =>
                    `<strong>${movie.title}</strong><br>` +
                    `<em>${xMetric}:</em> ${movie[xMetric]}<br>` +
                    `<em>${yMetric}:</em> ${movie[yMetric]}`
                ).join('<br><br>');

                tooltip.style("visibility", "visible")
                    .html(tooltipContent);
            })
            .on("mousemove", event => {
                tooltip.style("top", (event.pageY - 10) + "px")
                    .style("left", (event.pageX + 10) + "px");
            })
            .on("mouseout", () => {
                tooltip.style("visibility", "hidden");
            });
    }
}).catch(error => {
    console.error('Error loading the data:', error);
});
```

4.2. Prikaz konačnih funkcionalnosti i ponašanja

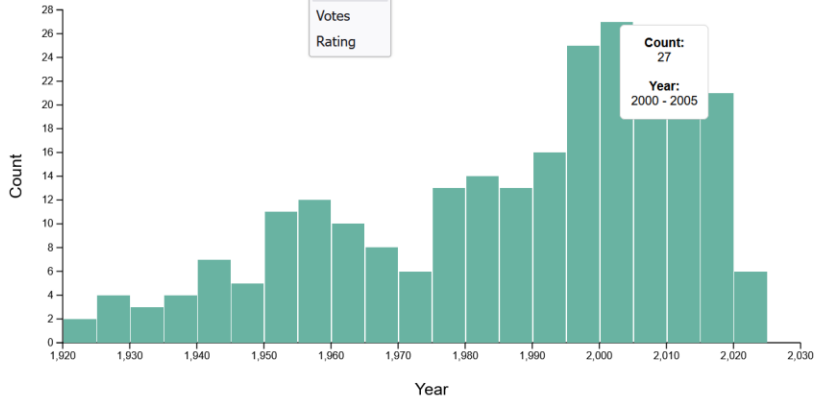


Home
Movie Collection
Data Distribution
Data Correlation
Top Directors

Data Distribution

Data: Year Genre: All Genres

Year
Duration
Votes
Rating

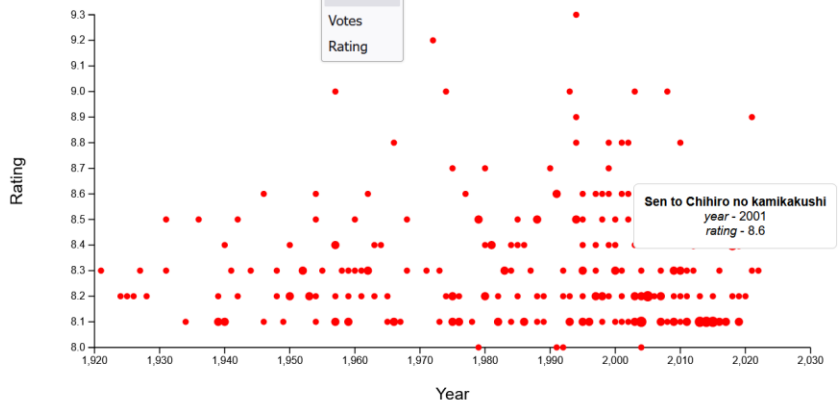


Home
Movie Collection
Data Distribution
Data Correlation
Top Directors

Data Correlation

X-axis: Year Y-axis: Rating

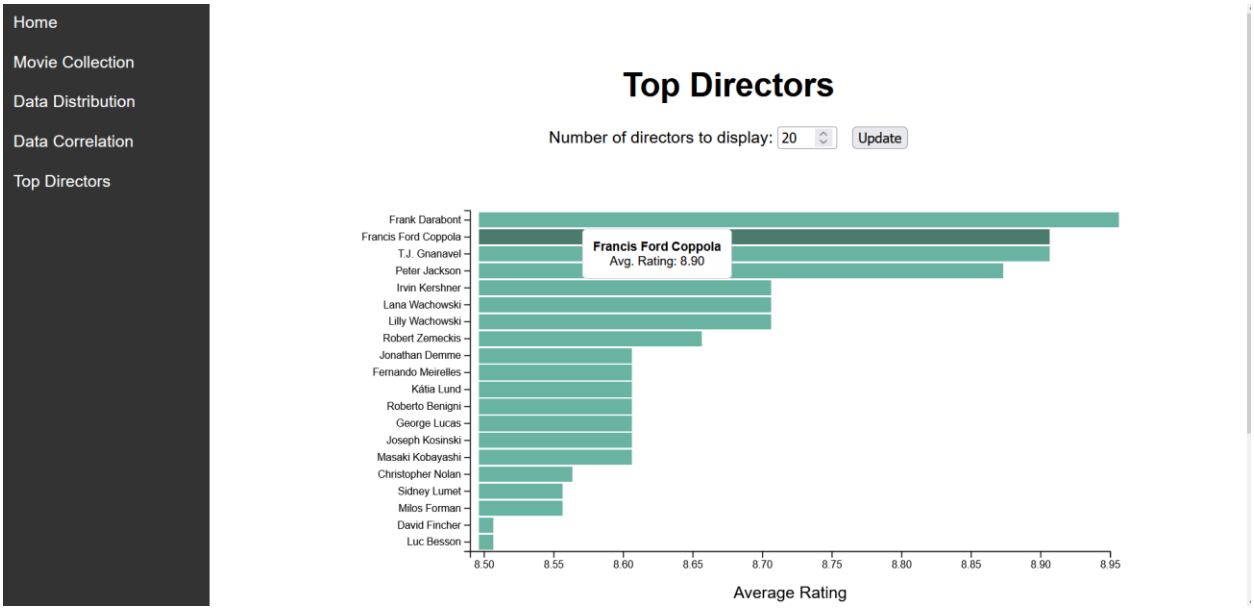
Year
Duration
Votes
Rating



5. KV5 - Dovršetak projektnog zadatka i pisanje dokumentacije

5.1. Eventualne preinake i dorade rješenja - u dogovoru s nastavnikom

U odnosu na prethodno rješenje, dodana je još jedna vizualizacija sa prikazom stupčastih dijagrama koji prikazuju poredak najuspješnijih direktora po prosječnoj ocjeni njihovih filmova. Uz to, uvedene su korekcije prethodnih vizualizacija kao što su: dinamično imenovanje X i Y osi grafova na temelju odabranih podataka, prikazivanje dodatnih informacija o podatcima kada pokazujemo na dijelove grafa sa mišem, centriranje i reformatiranje cijelog sadržaja dijagrama te njihovih tekstova, ispravci što se tiče preklapanja teksta i sl. Dodatno je još na početnu stranicu dodana slika sa prikazom formata json podataka koji su korišteni.

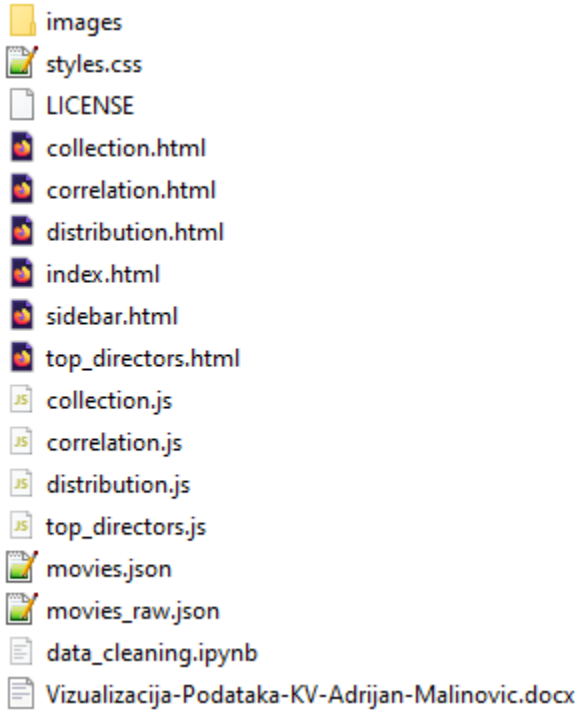


5.2. Izrada dokumenta - projektne dokumentacije

[U ovom zadatku potrebno je izraditi projektnu dokumentaciju koja će opisati proces i rezultate projektnog zadatka. Projektna dokumentacija obično uključuje opis projektnog zadatka, potrebne alate, proces rada, opis dizajna i vizualizacije podataka, izvješća o provedenim testovima i rezultatima, zaključak i slično. Cilj je da dokumentacija bude jasna, detaljna i potpuna kako bi drugi mogli razumjeti i koristiti vaše rješenje.]

Z-5.2.1. Hijerarhija projekta.

Datoteka *movies_raw.json* sadrži sirove podatke kako su preuzeti iz online izvora, oni se mogu očistiti pokretanjem Python bilježnice *data_cleaning.ipynp* kako bi dobilu čiste podatke *movies.json*. Svaka vizualizacija (pod stranica, ima ih 4) se sastoji od svojih HTML i JS datoteka istog imena radi lakšeg nadograđivanja i preinaka, a i radi čitljivosti. Početna stranica (*index.html*) nema svoju JS datoteku jer je njezina priroda statična, a imamo i posebnu datoteku za navigaciju podstranica jer se ona uključuje u svako do podstranica (*sidebar.html*). Svi CSS stilovi se nalaze u jednoj datoteci (*styles.css*).



Z-5.2.2. Popis korištenih tehnologija, bez opisa.

- HTML, JS, CSS, Python
- D3, Pandas
- Visual Studio Code, Google Colab

Z-5.2.3. Upute za postavljanje.

- Otići na GitHub repozitorij projekta te ga klonirati na proizvoljno mjesto
- Navigirati u folder gdje je spremljen projekt te pokrenuti naredbu python –m http.server 8000
- Otići u web preglednik na adresu localhost:8000

Z-5.2.4. Upute za korištenje.

Otvaranjem web stranice dolazimo na početnu stranicu, tu vidimo osnovne informacije o projektu i podatkovnom skupu, a imamo i sliku sa primjerom podatka iz JSON datoteke koji se koriste za vizualizaciju. Uz to vidimo i

poveznicu na GitHub repozitorij projekta. U naviganici na lijevoj stranici vidimo izbornik sa 4 tipa vizualizacije korištenih podataka. Klikom na svaki od njih nas stranica vodi do pripadajućih prikaza. Tu se korisnik može koristiti padajućim izbornicima za podešavanje/sortiranje raznih prikaza, kako bi vidio informacije koje ga zanimaju, na većinu stvari se može prijeći pokazivačem miša za dodatne informacije, a neki dijelovi se mogu i kliknuti.

Literatura

https://d3js.org/

https://www.w3schools.com/html/

https://www.w3schools.com/js/

https://datavizcatalogue.com/

Prilog

https://github.com/adrijanmalinovic/Vizualizacija-Podataka-KV