



University of Agriculture

Faisalabad

CS-506 (Big DATA Analysis)

Documentation for Data Analysis

Web Application

Warda Adan

2022-ag-7742

Degree: BSCS 6TH A

Submitted to: Sir Ahmad

Abstract

This web application is an interactive Data Analysis Dashboard designed to facilitate seamless data upload, preprocessing, exploratory analysis, and visualization. It provides users with an intuitive interface to upload datasets (CSV or Excel), select various machine learning algorithms (supervised, unsupervised, deep learning, graph algorithms), and generate insightful visualizations and metrics. The platform supports attribute selection, algorithm configuration, and dynamic visualization of results through interactive graphs, trees, and network representations. Built with modern web technologies, including HTML, CSS, JavaScript, and Chart.js, this dashboard offers an accessible environment for data scientists, analysts, and students to experiment with data analysis workflows directly in the browser.

Documentation for Data Analysis	1
Web Application.....	1
Abstract	2
1. Introduction	5
2. Features & Capabilities	5

3. Technology Stack	6
4. User Interface Overview	6
5. Functional Workflow.....	6
6. Code Breakdown & Explanation.....	7
6.1 HTML Structure	7
6.2 CSS Styling.....	8
6.3 JavaScript Functionality	8
Variables & DOM References	8
Event Listeners	8
Data Upload & Parsing	9
Data Handling & Preview	9
Analysis & Visualization.....	9
Helper Functions	9
7. Usage Instructions.....	10
8. Potential Enhancements.....	10
9. Conclusion	10
Code Explanation in Detail.....	11
1. HTML Structure	11
2. CSS Styling	11
3. JavaScript Functionality	11
HTML Structure.....	12
Inside <head>	13
Inside <body>	14
Header Section.....	14
Data Upload Section	14
Loading Indicator.....	15
Results Section	15
Inside Results Section.....	16
Footer	16

2. CSS Styles	17
3. JavaScript Functionality	18
3.1 Variable Declarations	18
3.2 Event Listeners	19
3.3 File Handling & Parsing	20
3.4 Handling Parsed Data	22
3.5 Data Preview	23
3.6 Running Analysis	23
3.7 Displaying Results	24
3.8 Helper Functions	25
Summary	25
Final Notes	26

1. Introduction

This dashboard serves as a comprehensive frontend tool enabling users to upload datasets, select analysis algorithms, visualize data, and observe simulated metrics—all within a single, interactive web page. It abstractly mimics the core functionalities of a data analysis pipeline, making it suitable for educational demonstrations, prototyping, or lightweight exploratory analysis.

2. Features & Capabilities

- **File Upload & Parsing:** Supports CSV and Excel files with drag-and-drop functionality.
- **Data Preview:** Shows a snapshot of uploaded data for initial review.
- **Attribute Selection:** Allows users to select attributes for analysis and visualization.
- **Algorithm Selection:** Provides options across supervised, unsupervised, deep learning, and graph algorithms.
- **Visualization Options:** Multiple graph types (scatter, line, bar, pie, area, tree, heatmap, network).
- **Dynamic Visualization Rendering:** Generates charts and textual representations based on selected algorithms and graph types.
- **Metrics Display:** Shows simulated performance metrics for selected algorithms.
- **Responsive Design:** Optimized for various screen sizes.
- **Interactive UI:** Includes tabs, hover effects, and clickable options for an engaging experience.

3. Technology Stack

- **HTML5:** Structure and semantic layout.
- **CSS3:** Styling with custom variables, responsive design, and visual effects.
- **JavaScript:** Functionality, data processing, and interaction handling.
- **Chart.js:** For rendering various charts and graphs.
- **PapaParse:** CSV parsing.
- **XLSX.js:** Excel file reading.
- **DOM Manipulation:** For dynamic content updates and interactivity.

4. User Interface Overview

The dashboard contains the following main sections:

- **Header:** Title and subtitle.
- **Data Upload Section:** File selection, drag-and-drop, and process button.
- **Loading Indicator:** Visual feedback during data processing.
- **Results Section:** Hidden initially; displays data preview, algorithm options, visualization options, attribute selection, analysis controls, and results.
- **Footer:** Credits and ownership info.

5. Functional Workflow

1. Upload Data:

- a. Drag/drop or select CSV/XLSX file.

- b. File parsed, and data preview displayed.
- 2. Data Preview & Attribute Selection:**
 - a. Show a snapshot of data.
 - b. Populate dropdowns for target attribute.
 - c. Select attributes for visualization.
- 3. Algorithm & Visualization Selection:**
 - a. Choose algorithms across categories.
 - b. Choose visualization types.
 - c. Select target column if needed.
 - d. Pick framework for deep learning.
- 4. Run Analysis:**
 - a. Generate simulated metrics.
 - b. Render charts, trees, network diagrams based on selections.
- 5. View Results:**
 - a. Metrics and visualizations displayed interactively.

6. Code Breakdown & Explanation

6.1 HTML Structure

- **Container & Layout:**
 - All content wrapped inside `.container`.
 - Header section contains titles and subtitles.
 - Upload section for data upload.
 - Results section hidden initially, shown after data upload.
- **Upload Section:**
 - File input (`<input type="file">`) hidden, styled as a drag-and-drop zone.
 - Button for processing data.
- **Results Section:**
 - Data preview table.

- Algorithm selection with tabs.
- Graphical options.
- Attribute selection.
- Target column selector.
- Analysis button.
- Visualization and metrics sections.
- **Footer:**
 - Contains copyright and attribution.

6.2 CSS Styling

- **Colors & Variables:**
 - Uses CSS variables for consistent theming.
- **Layout & Responsiveness:**
 - Grid and flexbox layouts adapt to screen sizes.
- **UI Elements:**
 - Buttons, cards, hover effects, tabs, and tooltips for interactivity.
- **Animations:**
 - Fade-in effects for results.
- **Responsive Design:**
 - Media queries adjust layout for smaller screens.

6.3 JavaScript Functionality

Variables & DOM References

- Variables hold data, selections, and current charts.
- DOM elements are referenced for event handling and updates.

Event Listeners

- File input change & drag/drop for file upload.

- Button clicks for processing and analysis.
- Tab clicks for switching algorithm categories.
- Algorithm & graph option selection toggles.

Data Upload & Parsing

- Supports CSV via PapaParse.
- Supports Excel via XLSX.js.
- After parsing, data is stored, and UI is updated.

Data Handling & Preview

- Identifies numeric columns.
- Populates attribute options and target selector.
- Displays a data preview table.

Analysis & Visualization

- Simulates metrics with random data.
- Creates various chart types using Chart.js.
- Generates textual representations for tree and network graphs.
- Handles multiple visualization options simultaneously.

Helper Functions

- Formatting algorithm & graph names.
- Generating sample data for charts.
- Assigning colors based on algorithms.
- Creating tree and network textual diagrams.

7. Usage Instructions

1. Upload your dataset by selecting a CSV or Excel file.
2. Click "Process Data" to parse and view data preview.
3. Select attributes for visualization.
4. Choose algorithms from each category.
5. Pick desired graph types.
6. Specify target column if using supervised algorithms.
7. Click "Run Analysis" to generate metrics and visualizations.
8. Explore generated charts, trees, and network diagrams.

8. Potential Enhancements

- Backend integration for real data analysis.
- Support for more algorithms and models.
- Upload progress indicators.
- Export options for charts and metrics.
- User authentication and session management.
- Data filtering and transformation tools.

9. Conclusion

This web-based Data Analysis Dashboard acts as an educational and prototyping tool, demonstrating core concepts in data science workflows. Its interactive UI, dynamic visualizations, and simulation of analysis metrics provide users with an engaging environment to explore data analysis techniques directly in their browsers.

Code Explanation in Detail

1. HTML Structure

The HTML code sets up the webpage layout, including headers, upload area, results display, and footer. It uses semantic tags and classes for styling and scripting hooks.

2. CSS Styling

- Implements a dark-themed aesthetic with vibrant accents.
- Utilizes CSS variables for easy theme management.
- Defines styles for all UI components, responsive behavior, hover effects, and animations.

3. JavaScript Functionality

- **File Handling:**
 - Listens for file input or drag/drop.
 - Uses PapaParse and XLSX.js to parse CSV and Excel files.
 - Stores parsed data globally (uploadedData).
- **Data Processing & Preview:**
 - Identifies numeric columns for attribute selection.
 - Populates dropdowns and checkboxes dynamically.
 - Shows a preview table of the first few rows.
- **Algorithm & Graph Selection:**
 - Users select algorithms and graph types, toggling classes for visual feedback.
 - Selections stored in arrays.
- **Analysis & Visualization:**
 - Simulates metrics with random values.
 - Uses Chart.js to generate charts (scatter, line, bar, pie, area, heatmap).

- Creates textual representations for tree and network graphs.
- **Interactivity & UI Updates:**
 - Tabs switch algorithm categories.
 - Attribute checkboxes update selected attributes.
 - Buttons trigger data processing and analysis.
- **Utilities:**
 - Formatting names.
 - Generating sample data.
 - Assigning colors based on algorithm type.

Certainly! Here's a detailed, line-by-line explanation of the entire code, covering HTML, CSS, and JavaScript sections. This will help you understand how each part functions and how they fit together to create the interactive dashboard.

HTML Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
  <!-- Metadata and external resources -->
</head>
<body>
  <!-- Main container holding all content -->
</body>
```

`</html>`

- `<!DOCTYPE html>`: Declares the document type and version, informing browsers this is an HTML5 document.
- `<html lang="en">`: Root element with language set to English.
- `<head>`: Contains metadata, links to stylesheets, and scripts.
- `<body>`: The main visible content of the webpage.

Inside `<head>`

`<meta charset="UTF-8">`

`<meta name="viewport" content="width=device-width, initial-scale=1.0">`

`<title>Data Analysis Proje</title>`

`<!-- Font Awesome for icons -->`

`<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">`

`<!-- Internal CSS styles -->`

`<style>`

`/* CSS code here */`

`</style>`

- `<meta charset="UTF-8">`: Sets character encoding to support special characters.
- `<meta name="viewport" ...>`: Ensures responsiveness on mobile devices.
- `<title>`: Page title shown in browser tab.
- `<link>`: Loads Font Awesome icons via CDN.
- `<style>`: Contains all CSS styling rules.

Inside <body>

```
<div class="container">
  <!-- Header, upload, results, footer -->
</div>
```

- <div class="container">: Wraps all main content for layout purposes.

Header Section

```
<header>
  <h1>2022-ag-7742</h1>
  <h1>Data Analysis Dashboard</h1>
  <p class="subtitle">Upload your dataset, select algorithms, and
visualize insights with multiple interactive graphs</p>
</header>
```

- <header>: Semantic tag for page header.
- Two <h1> tags: Titles.
- <p class="subtitle">: Subtitle paragraph.

Data Upload Section

```
<div class="upload-section">
  <h2 class="section-title"><i class="fas fa-cloud-upload-alt"></i>
Data Upload</h2>
  <div class="form-group">
```

```

    <label for="file-upload">Select your dataset (CSV or
Excel)</label>
    <input type="file" id="file-upload" accept=".csv,.xlsx,.xls">
    <div class="file-upload" id="file-upload-label">
        <!-- Icon and instructions -->
    </div>
</div>
<button id="upload-btn" class="btn" disabled>
    <i class="fas fa-cogs"></i> Process Data
</button>
</div>

```

- <input type="file">: Hidden file input for uploading files.
- <div class="file-upload">: Styled area for drag-and-drop or click.
- <button>: Starts data processing, disabled until file selected.

Loading Indicator

```

<div class="loading" id="loading-indicator">
    <div class="spinner"></div>
    <p>Processing your data. This may take a moment...</p>
</div>

```

- Hidden by default; shown during data processing.

Results Section

```

<div class="results-section" id="results-section">
    <!-- Data preview, algorithms, visualization, metrics, etc. -->

```

</div>

- Contains all the outputs after data upload.

Inside Results Section

- **Data Preview**

```
<div class="data-preview">
```

```
  <h2 class="section-title"><i class="fas fa-table"></i> Data  
  Preview</h2>
```

```
    <div id="data-table-container"></div>  
</div>
```

- Placeholder for displaying a data table.
- **Algorithm Selection**

```
<div class="options-section">
```

```
  <!-- Algorithm options, visualization options, attribute selection,  
  target column, analyze button, visualizations, metrics -->  
</div>
```

- Multiple nested blocks for different selections, each styled as cards.

Footer

```
<footer>
```

```
  <p>© 2025 Advanced Data Analysis Dashboard | Powered by Warda  
  Adan</p>
```


</footer>

- Contains credits and copyright.

2. CSS Styles

The <style> block contains all styling rules, which define the look and feel of elements:

- **Color variables (:root):** Centralized theme colors.
- **Reset styles (*):** Remove default margin and padding, set box-sizing.
- **Body:** Sets font-family, background gradient, text color, min-height.
- **Container:** Max width, centered, with padding.
- **Header:** Text alignment, margin, positioning.
- **Decorative pseudo-element (header::after):** Adds a gradient underline.
- **Headings (h1):** Large font, gradient text.
- **Subtitle:** Slight opacity.
- **Upload section:** Transparent background, rounded corners, hover effects.
- **File upload area:** Dashed border, hover animations, background effects.
- **Buttons (.btn):** Gradient background, hover animations, icons.
- **Results & visualization sections:** Hidden initially, animated fade-in.
- **Tables:** Styled with gradient headers, hover effects.
- **Responsive adjustments:** Media queries for smaller screens.

3. JavaScript Functionality

The <script> blocks contain all the interactive behavior.

3.1 Variable Declarations

```
// Global variables
```

```
let uploadedData = null;  
let selectedAlgorithms = [];  
let selectedGraphs = [];  
let selectedAttributes = [];  
let currentCharts = [];  
let columnNames = [];  
let numericColumns = [];
```

- Store uploaded data, selected options, and chart instances.

```
// DOM element references
```

```
const fileUpload = document.getElementById('file-upload');  
const fileUploadLabel = document.getElementById('file-upload-label');  
const fileName = document.getElementById('file-name');  
const uploadBtn = document.getElementById('upload-btn');  
const loadingIndicator = document.getElementById('loading-indicator');  
const resultsSection = document.getElementById('results-section');  
const algorithmOptions = document.getElementById('algorithm-  
options');  
const graphOptions = document.getElementById('graph-options');  
const attributeOptions = document.getElementById('attribute-options');  
const targetColumnSelect = document.getElementById('target-column');  
const frameworkSelection = document.getElementById('framework-  
selection');
```

```
const analyzeBtn = document.getElementById('analyze-btn');
const graphsContainer = document.getElementById('graphs-container');
const metricsContainer = document.getElementById('metrics-
container');
const dataTableContainer = document.getElementById('data-table-
container');
```

- References to key DOM elements for event wiring and updates.

3.2 Event Listeners

```
fileUpload.addEventListener('change', handleFileUpload);
fileUploadLabel.addEventListener('click', () => fileUpload.click());
uploadBtn.addEventListener('click', processUploadedFile);
analyzeBtn.addEventListener('click', runAnalysis);
```

- React to file selection, upload button, and analyze button clicks.

// Drag-and-drop events

```
fileUploadLabel.addEventListener('dragover', (e) => {...});
fileUploadLabel.addEventListener('dragleave', (e) => {...});
fileUploadLabel.addEventListener('drop', (e) => {...});
```

- Enable drag-and-drop file upload with visual cues.

// Tab switching for algorithm categories

```
document.querySelectorAll('.tab').forEach(tab => {...});
```

- Switches between categories: supervised, unsupervised, deep learning, graph.

```
// Algorithm option toggles
document.querySelectorAll('.option-card[data-algo]').forEach(card =>
{...});
```

- Adds/removes algorithms from the selected list.

```
// Graph options toggles
document.querySelectorAll('#graph-options .option-card').forEach(card
=> {...});
```

- Adds/removes graph types.

3.3 File Handling & Parsing

```
function handleFileUpload(event) {
  const file = event.target.files[0];
  if (file) {
    fileName.textContent = file.name;
    uploadBtn.disabled = false;
  } else {
    fileName.textContent = 'No file selected';
    uploadBtn.disabled = true;
  }
}
```

- Updates filename display and enables the process button when a file is selected.

```
function processUploadedFile() {
  const file = fileUpload.files[0];
  if (!file) return;
```

```

loadingIndicator.style.display = 'block';
resultsSection.style.display = 'none';

const fileExtension = file.name.split('.').pop().toLowerCase();

if (fileExtension === 'csv') {
    parseCSV(file);
} else if (fileExtension === 'xlsx' || fileExtension === 'xls') {
    parseExcel(file);
} else {
    alert('Please upload a valid CSV or Excel file.');
```

loadingIndicator.style.display = 'none';

```

}
}

```

- Determines file type and calls appropriate parser.

```

function parseCSV(file) {
    Papa.parse(file, {
        header: true,
        complete: function(results) {
            handleParsedData(results.data, results.meta.fields);
        },
        error: function(error) {...}
    });
}

```

- Parses CSV, calls handleParsedData with parsed data.

```

function parseExcel(file) {
    const reader = new FileReader();
    reader.onload = function(e) {

```

```

    try {
        const data = new Uint8Array(e.target.result);
        const workbook = XLSX.read(data, { type: 'array' });
        // Get first sheet and convert to JSON
    } catch (error) {...}
};
reader.onerror = function(error) {...}
reader.readAsArrayBuffer(file);
}

```

- Reads Excel file, converts to JSON format.

3.4 Handling Parsed Data

```

function handleParsedData(data, columns) {
    if (!data || !columns || columns.length === 0) {...}
    uploadedData = data;
    columnNames = columns;
    selectedAttributes = [];
    // Identify numeric columns
    numericColumns = identifyNumericColumns(data, columns);
    // Populate target column select
    populateTargetColumn(columns);
    // Populate attribute checkboxes
    populateAttributes(columns);
    // Show data preview
    displayDataPreview(data, columns);
    // Hide loading, show results
}

```

- Stores data, identifies numeric columns, updates UI.

3.5 Data Preview

```
function displayDataPreview(data, columns) {
  const previewData = data.slice(0, 10);
  let tableHTML = '<table><thead><tr>';
  columns.forEach(column => {
    tableHTML += '<th>${column}</th>';
  });
  tableHTML += '</tr></thead><tbody>';
  previewData.forEach(row => {
    tableHTML += '<tr>';
    columns.forEach(column => {
      const value = row[column] !== undefined ? row[column] : "";
      tableHTML += '<td>${typeof value === 'object' ?
JSON.stringify(value) : value}</td>';
    });
    tableHTML += '</tr>';
  });
  tableHTML += '</tbody></table>';
  dataTableContainer.innerHTML = tableHTML;
}
```

- Creates an HTML table for a snapshot of data.

3.6 Running Analysis

```
function runAnalysis() {
  // Validates selections
  // Shows loading indicator
```

```

// Simulates async processing with setTimeout
setTimeout(() => {
  displayResults();
  loadingIndicator.style.display = 'none';
}, 2000);
}

```

- After delay, calls displayResults() to generate visualizations.

3.7 Displaying Results

```

function displayResults() {
  // Destroy previous charts
  // Clear graph and metrics containers

  // For each algorithm, generate metrics (simulated)
  selectedAlgorithms.forEach(algorithm => {
    const metrics = generateAlgorithmMetrics(algorithm);
    // Create metric card with color depending on algorithm type
    metricsContainer.appendChild(metricCard);
  });

  // For each selected graph type, generate visualizations
  selectedGraphs.forEach(graphType => {
    selectedAlgorithms.forEach(algorithm => {
      // Create graph card
      // Generate data based on type and algorithm
      // For special graphs (tree, network), generate ASCII-like
      diagrams
      // For others, generate Chart.js charts
    });
  });
}

```



```
});  
}
```

- Shows metrics and visualizations with dynamic content.

3.8 Helper Functions

- **generateAlgorithmMetrics**: Creates random metrics for algorithms.
- **formatAlgorithmName, formatGraphName**: Human-readable labels.
- **getGraphIcon, getChartType**: Map graph types to icons and chart types.
- **generateChartData**: Generate sample datasets for charts.
- **generateTreeGraph**: Creates a textual tree diagram based on attributes.
- **generateNetworkGraph**: Creates a textual network diagram.
- **generateHeatmapData**: Random importance scores for heatmaps.
- **Color functions**: Assign consistent colors per algorithm.

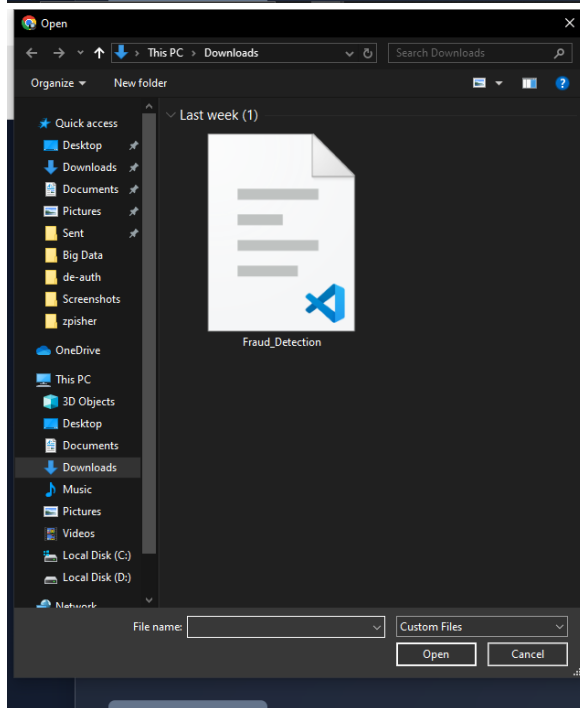
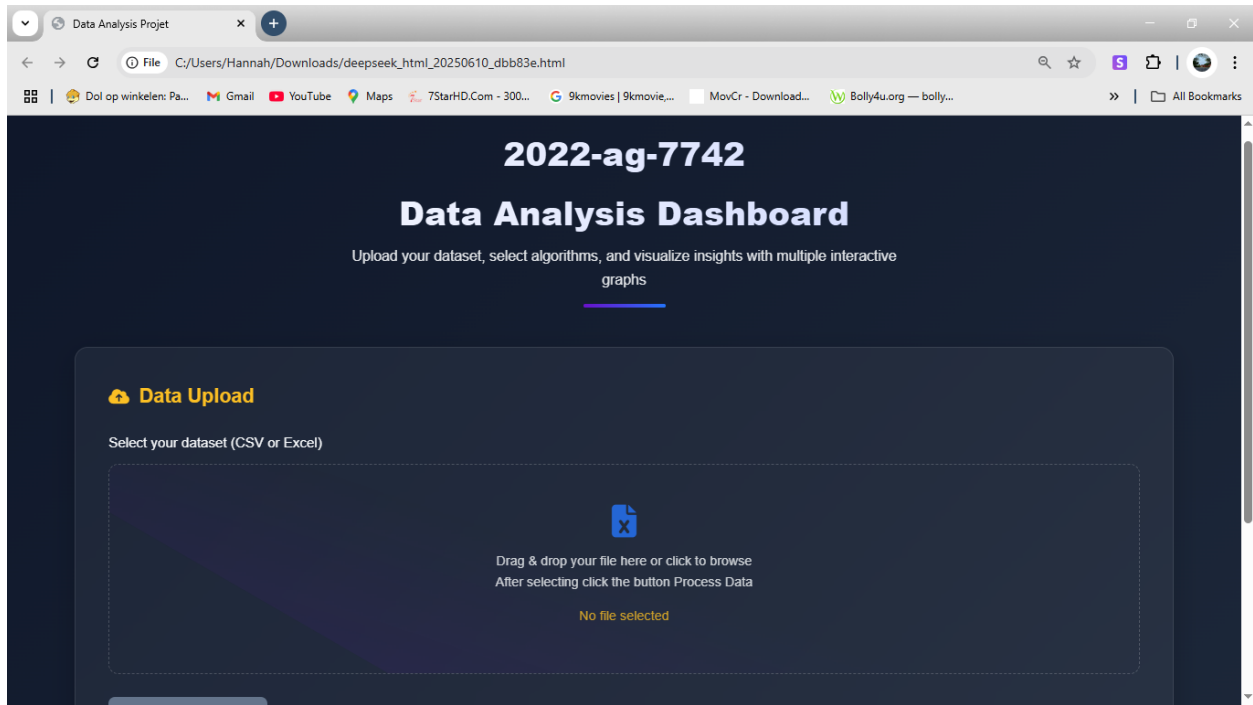
Summary

This detailed explanation covers each line and concept in the code, showing how the webpage is built, styled, and made interactive. The HTML structures the content, CSS styles it beautifully, and JavaScript brings it all to life with data processing, user interaction, and dynamic visualization.

Final Notes

This dashboard provides a foundational framework for interactive data analysis in a web environment. It is primarily a frontend prototype, with simulated analysis results, suitable for demonstrations, learning, and initial exploration before integrating with backend data processing pipelines.

Screenshots of my website



Data Analysis Project

C:/Users/Hannah/Downloads/deepseek_html_20250610_dbb83e.html

Dol op winkelen: Pa... Gmail YouTube Maps 7StarHD.Com - 300... 9kmovies | 9kmovie... MovCr - Download... Bolly4u.org — bolly...

Process Data

Data Preview

Transaction_ID	User_ID	Transaction_Amount	Transaction_Type	Time_of_Transaction	Device_Used	Location	Previous_Fraudulent_Transactions
T1	4174	1292.76	ATM Withdrawal	16.0	Tablet	San Francisco	0
T2	4507	1554.58	ATM Withdrawal	13.0	Mobile	New York	4
T3	1860	2395.02	ATM Withdrawal		Mobile		3
T4	2294	100.1	Bill Payment	15.0	Desktop	Chicago	4
T5	2130	1490.5	POS Payment	19.0	Mobile	San Francisco	2
T6	2095	2372.04	ATM Withdrawal	15.0	Desktop	Boston	3

Data Analysis Project

C:/Users/Hannah/Downloads/deepseek_html_20250610_dbb83e.html

Dol op winkelen: Pa... Gmail YouTube Maps 7StarHD.Com - 300... 9kmovies | 9kmovie... MovCr - Download... Bolly4u.org — bolly...

Algorithm Selection

Choose one or more algorithms to analyze your data

Supervised

Unsupervised

Deep Learning

Graph

Linear Regression

Predict continuous values based on relationships

Logistic Regression

Binary classification algorithm

Decision Tree

Tree-like model for classification/regression

Random Forest

Ensemble of decision trees

Support Vector Machine

Powerful classification algorithm

Neural Network

Multi-layer perceptron for complex patterns

Visualization Options

Select multiple graph types to visualize your results

Scatter Plot

Show relationships between variables

Line Chart

Visualize trends over time

Bar Chart

Compare different categories

Pie Chart

Show proportional data

Area Chart

Show cumulative data

Tree Graph

Show hierarchical data

Heatmap

Visualize matrix data

Network Graph

Show connections between nodes

Data Analysis Project

C:/Users/Hannah/Downloads/deepseek_html_20250610_dbb83e.html

Supervised Unsupervised Deep Learning Graph

K-Means Clustering
Unsupervised clustering algorithm

Principal Component Analysis
Dimensionality reduction technique

DBSCAN
Density-based clustering algorithm

Gaussian Mixture Model
Probabilistic clustering approach

Scatter Plot
Show relationships between variables

Line Chart
Visualize trends over time

Bar Chart
Compare different categories

Pie Chart
Show proportional data

Area Chart
Show cumulative data

Tree Graph
Show hierarchical data

Heatmap
Visualize matrix data

Network Graph
Show connections between nodes

Attribute Selection

Select attributes to include in visualizations

☐ Transaction_ID ☒ User_ID ☒ Transaction_Amount ☐ Transaction_Type ☐ Time_of_Transaction ☐ Device_Used ☐ Location ☒ Previous_Fraudulent_Transaction

Data Analysis Project

C:/Users/Hannah/Downloads/deepseek_html_20250610_dbb83e.html

☐ Account_Age ☐ Number_of_Transactions_Last_24H ☐ Payment_Method ☐ Fraudulent

Select Target Column (for supervised learning)

-- Select Target Column --

Transaction_ID

User_ID

Transaction_Amount

Transaction_Type

Time_of_Transaction

Device_Used

Location

Previous_Fraudulent_Transactions

Account_Age

Number_of_Transactions_Last_24H

Payment_Method

Fraudulent

Performance metrics for your selected algorithms

Analysis results will appear here

Data Analysis Project

C:/Users/Hannah/Downloads/deepseek_html_20250610_dbb83e.html

Account_Age Number_of_Transactions Previous_Fraudulent_Transactions

Select Target Column (for supervised learning)

Previous_Fraudulent_Transactions

Run Analysis

Visualizations

Your selected visualizations will appear below

Select algorithms and graph types to generate visualizations

Model Metrics

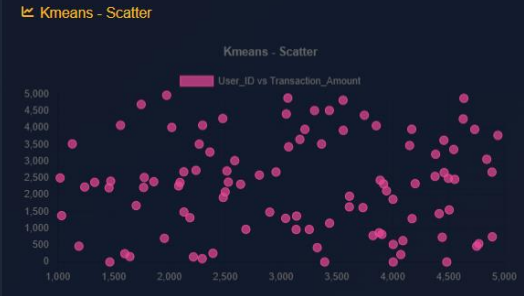
Performance metrics for your selected algorithms

Analysis results will appear here

Data Analysis Project

C:/Users/Hannah/Downloads/deepseek_html_20250610_dbb83e.html

Kmeans - Scatter



Model Metrics

Performance metrics for your selected algorithms

Model	Score
Kmeans	98.49%
Silhouette Score	0.63