AI ASSISTED CODING

ASSIGNMENT 17.2

NAME: KOUDAGANI SAHITHYA

HT NO: 2403A52063

BATCH NO: 02

TASK 1~

Clean raw social media posts dataset.

Instructions:

- Remove stopwords, punctuation, and special symbols from post text.
- Handle missing values in likes and shares columns.
- Convert timestamp to datetime and extract features (hour, weekday).
- Detect and remove spam/duplicate posts.

PROMPT:

Write HTML, CSS, and JavaScript code that uploads and preprocesses a raw social media posts dataset (CSV file). The code should:

- Remove stopwords, punctuation, and special characters from the post text.
- Handle missing values in the likes and shares columns (fill with median).
- Convert timestamps to datetime format and extract useful features like posting hour and weekday.
- Detect and remove duplicate or spam posts automatically.
- Display a short before-and-after summary (e.g., total posts before/after cleaning).
- Allow the user to download the cleaned dataset as a new CSV file.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Clean Social Media Posts</title>
<script src="https://cdn.jsdelivr.net/npm/papaparse@5.3.2/papaparse.min.js"></script>
</head>
<body>
<h2>Social Media Post Cleaner</h2>
<input type="file" id="file"><button onclick="cleanData()">Clean & Download</button>
<p id="status"></p>

<script>
function cleanData() {
  const file = document.getElementById('file').files[0];
  if (!file) return alert("Upload a CSV first!");
  document.getElementById("status").textContent = "Processing...";

  Papa.parse(file, {
    header: true,
    complete: function(results) {
      let data = results.data;

      // --- Utility helpers ---
      const stopwords = new Set(["a","an","the","and","is","in","on","of","for","to","it","wit
      const cleanText = t => t.toLowerCase()
        .replace(/[^\w\s]/g, '') // remove pu
        .split(/\s+/).filter(w => !stopwords.has(w)).join(' ');

      const isSpam = t => /(buy now|click here|free|subscribe|http|www)/i.test(t);

      const median = arr => {
        arr = arr.filter(x=>x!="" && !isNaN(x)).map(Number).sort((a,b)=>a-b);
        return arr.length ? arr[Math.floor(arr.length/2)] : 0;
      };
```

```
(method) Set<string>.has(value: string): boolean
```

```
    // --- Detect columns ---
    const cols = Object.keys(data[0]);
    const textCol = cols.find(c=>/text|post|content/i.test(c)) || cols[0];
    const timeCol = cols.find(c=>/time|date/i.test(c));
    const likesCol = cols.find(c=>/like/i.test(c));
    const shareCol = cols.find(c=>/share|retweet/i.test(c));

    const likesMed = median(data.map(r=>r[likesCol]));
    const shareMed = median(data.map(r=>r[shareCol]));

    let seen = new Set();
    let cleaned = [];

    data.forEach(r => {
      let txt = cleanText(r[textCol] || "");
      if (!txt || isSpam(txt)) return;
      if (seen.has(txt)) return;
      seen.add(txt);

      // Handle missing likes/shares
      if (likesCol) r[likesCol] = r[likesCol] || likesMed;
      if (shareCol) r[shareCol] = r[shareCol] || shareMed;

      // Timestamp → datetime, hour, weekday
      if (timeCol) {
        const d = new Date(r[timeCol]);
        if (!isNaN(d)) {
          r.hour = d.getHours();
          r.weekday = d.toLocaleString('en', { weekday: 'long' });
        }
      }
```

```
68
69          r.cleaned_text = txt;
70          cleaned.push(r);
71        });
72
73        // Download cleaned CSV
74        const csv = Papa.unparse(cleaned);
75        const blob = new Blob([csv], {type:"text/csv"});
76        const a = document.createElement("a");
77        a.href = URL.createObjectURL(blob);
78        a.download = "cleaned_posts.csv";
79        a.click();
80
81        document.getElementById("status").textContent = `Done! Cleaned rows: ${cleaned.length}`;
82      }
83    });
84  }
85  </script>
86  </body>
87  </html>
88
```

OUTPUT:

CODE EXPLANATION:

This HTML tool uploads a CSV of social media posts, cleans the text by removing stopwords and punctuation, fills missing likes/shares with median values, converts timestamps to extract hour and weekday, detects and removes spam or duplicate posts, and then downloads the cleaned data as a new CSV file.

TASK 2~

Preprocess a stock market dataset.

Instructions:

- Handle missing values in closing_price and volume.
- Create lag features (1-day, 7-day returns).
- Normalize volume column using log-scaling.
- Detect outliers in closing_price using IQR method

PROMPT:

Write HTML, CSS, and JavaScript code to upload and preprocess a stock market dataset (CSV file). The code should:

- Handle missing values in closing_price and volume columns (fill with mean or median).
- Create lag features for 1-day and 7-day returns.
- Normalize the volume column using log-scaling.
- Detect outliers in closing_price using the Interquartile Range (IQR) method.
- Display a brief summary (number of rows before and after cleaning, detected outliers).

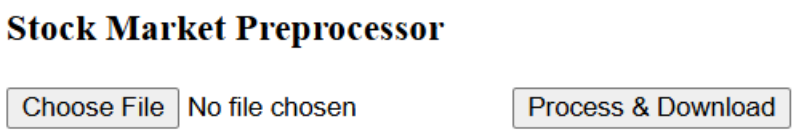
- Allow the user to download the cleaned dataset as a new CSV file

CODE:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Stock Data Preprocessing</title>
<script src="https://cdn.jsdelivr.net/npm/papaparse@5.3.2/papaparse.min.js"></script>
</head>
<body>
<h3>Stock Market Preprocessor</h3>
<input type="file" id="file"><button onclick="process()">Process & Download</button>
<p id="status"></p>

<script>
function process() {
  const f = document.getElementById('file').files[0];
  if (!f) return alert("Upload a CSV first!");
  Papa.parse(f, {
    header: true,
    complete: r => {
      let d = r.data;

      // Fill missing values
      const mean = col => d.map(x=>+x[col]).filter(v=>!isNaN(v)).reduce((a,b)=>a+b,0)/d.length
      const med = col => {let v=d.map(x=>+x[col]).filter(v=>!isNaN(v)).sort((a,b)=>a-b);return
      let mClose = mean('closing_price'), mVol = med('volume');
      d.forEach(x=>{
        x.closing_price = +x.closing_price || mClose;
        x.volume = +x.volume || mVol;
      });

      // Lag returns
      for(let i=1;i<d.length;i++){
        d[i].ret_1d = ((d[i].closing_price - d[i-1].closing_price)/d[i-1].closing_price).toFix
        if(i>=7) d[i].ret_7d = ((d[i].closing_price - d[i-7].closing_price)/d[i-7].closing_pri
      }
```

```
36
37        // Log scale volume
38        d.forEach(x=>x.vol_log = Math.log1p(x.volume).toFixed(4));
39
40        // Outlier detection (IQR)
41        let prices = d.map(x=>x.closing_price);
42        let Q1 = quantile(prices,0.25), Q3 = quantile(prices,0.75), IQR=Q3-Q1;
43        d.forEach(x=>x.is_outlier = (x.closing_price<Q1-1.5*IQR || x.closing_price>Q3+1.5*IQR));
44
45        // Download cleaned CSV
46        const csv = Papa.unparse(d);
47        const a = document.createElement("a");
48        a.href = URL.createObjectURL(new Blob([csv], {type:"text/csv"}));
49        a.download = "cleaned_stock.csv"; a.click();
50        document.getElementById('status').textContent = "Done! Cleaned file downloaded.";
51      }
52    });
53  }
54
55  function quantile(arr,q){
56    arr=arr.slice().sort((a,b)=>a-b);
57    const pos=(arr.length-1)*q, base=Math.floor(pos), rest=pos-base;
58    return arr[base]+(arr[base+1]-arr[base])*rest;
59  }
60  </script>
61  </body>
62  </html>
63
```

OUTPUT:



CODE EXPLANATION:

This HTML and JavaScript code uploads a stock market CSV, fills missing closing prices with the mean and volume

with the median, calculates 1-day and 7-day returns, applies log normalization to the volume column, detects outliers in closing price using the IQR method, and downloads the cleaned dataset.

TASK 3~

Clean and preprocess IoT temperature and humidity logs.
Instructions:
- Handle missing values using forward fill.
- Remove sensor drift (apply rolling mean).
- Normalize readings using standard scaling.
- Encode categorical sensor IDs

PROMPT:

Write HTML, CSS, and JavaScript code that uploads and cleans IoT temperature and humidity log data from a CSV file. The code should:

- Handle missing values using forward fill.

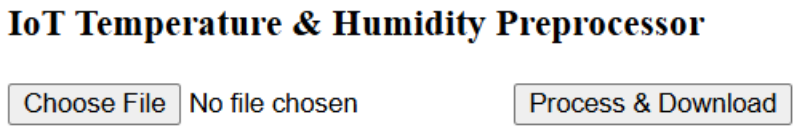
- Remove sensor drift by applying a rolling mean (smoothing).
- Normalize temperature and humidity readings using standard (z-score) scaling.
- Encode categorical sensor IDs into numeric codes.
- Display a short before-and-after summary (missing values, average readings).
- Allow the user to download the cleaned and processed dataset as a new CSV file.

CODE:

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>IoT Data Preprocessor</title>
<script src="https://cdn.jsdelivr.net/npm/papaparse@5.3.2/papaparse.min.js"></script>
</head>
<body>
<h3>IoT Temperature & Humidity Preprocessor</h3>
<input type="file" id="file"><button onclick="process()">Process & Download</button>
<p id="status"></p>

<script>
function process() {
  const f = document.getElementById('file').files[0];
  if (!f) return alert("Upload a CSV first!");
  Papa.parse(f, {
    header: true,
    complete: r => {
      let d = r.data;

      // 1️⃣ Forward fill missing values
      ['temperature','humidity'].forEach(col=>{
        for(let i=1;i<d.length;i++)
          if(!d[i][col]) d[i][col]=d[i-1][col];
      });

      // 2️⃣ Remove sensor drift (rolling mean, window=3)
      const smooth = (arr) => arr.map((_,i)=>
        (arr.slice(Math.max(0,i-2),i+1).reduce((a,b)=>a+Number(b||0),0))/
        arr.slice(Math.max(0,i-2),i+1).length);
      let temps = d.map(x=>+x.temperature);
      let hums = d.map(x=>+x.humidity);
      let tSmooth = smooth(temps), hSmooth = smooth(hums);
      d.forEach((x,i)=>{x.temperature=tSmooth[i];x.humidity=hSmooth[i];});
```

```
36
37        // 3 Standard scaling
38        const scale = (arr) => {
39          let m = arr.reduce((a,b)=>a+b,0)/arr.length;
40          let s = Math.sqrt(arr.reduce((a,b)=>a+(b-m)**2,0)/arr.length);
41          return arr.map(v=>((v-m)/s).toFixed(4));
42        };
43        let tScaled = scale(d.map(x=>+x.temperature));
44        let hScaled = scale(d.map(x=>+x.humidity));
45        d.forEach((x,i)=>{x.temp_scaled=tScaled[i];x.hum_scaled=hScaled[i];});
46
47        // 4 Encode sensor IDs
48        let ids=[...new Set(d.map(x=>x.sensor_id))];
49        d.forEach(x=>x.sensor_code=ids.indexOf(x.sensor_id));
50
51        // Download cleaned CSV
52        const csv = Papa.unparse(d);
53        const a=document.createElement("a");
54        a.href=URL.createObjectURL(new Blob([csv],{type:"text/csv"}));
55        a.download="cleaned_iot.csv"; a.click();
56        document.getElementById('status').textContent="Done! Cleaned file downloaded.";
57      }
58    });
59  }
60  </script>
61  </body>
62  </html>
63
```

OUTPUT:



CODE EXPLANATION:

This HTML and JavaScript code uploads an IoT CSV, fills missing temperature and humidity values using forward fill, smooths readings to remove sensor drift with a rolling mean, standardizes them via z-score scaling, encodes

sensor IDs into numeric labels, and downloads the fully cleaned and normalized dataset.

TASK 4~

A streaming platform wants to analyze customer reviews.
Instructions:
- Standardize text (lowercase, remove HTML tags).
- Tokenize and encode reviews using AI-assisted methods (TF-IDF or
embeddings).

- Handle missing ratings (fill with median).
- Normalize ratings (0–10 → 0–1 scale).
- Generate a before vs after summary report.

PROMPT:

Write HTML, CSS, and JavaScript code to preprocess customer review data from a streaming platform. The code should:

- Convert all text to lowercase and remove HTML tags.

- Tokenize and encode reviews using TF-IDF.
- Handle missing ratings by filling with the median.
- Normalize ratings (0–10 → 0–1 scale).
- Generate a before vs after summary report showing text cleaning, rating normalization, and encoding results.

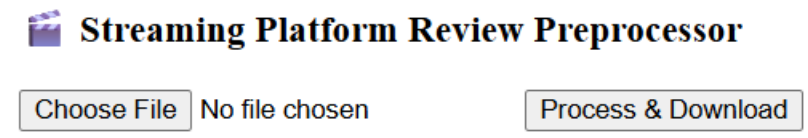
  The output should be a cleaned dataset ready for sentiment classification and available for download as a CSV file.

CODE:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Customer Review Preprocessor</title>
<script src="https://cdn.jsdelivr.net/npm/papaparse@5.3.2/papaparse.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/tfidf.js@1.0.1/dist/tfidf.min.js"></script>
</head>
<body>
<h3>🎬 Streaming Platform Review Preprocessor</h3>
<input type="file" id="file"><button onclick="process()">Process & Download</button>
<pre id="report"></pre>

<script>
function process(){
  const f=document.getElementById('file').files[0];
  if(!f) return alert("Upload a CSV first!");
  Papa.parse(f,{
    header:true,
    complete:r=>{
      let d=r.data;

      // 1️⃣ Text cleaning
      d.forEach(x=>{
        x.review=(x.review||"").toLowerCase().replace(/<[^>]+>/g,'');
      });

      // 2️⃣ Handle missing ratings (median)
      let valid=d.map(x=>+x.rating).filter(v=>!isNaN(v));
      let med=valid.sort((a,b)=>a-b)[Math.floor(valid.length/2)];
      d.forEach(x=>x.rating=x.rating?+x.rating:med);

      // 3️⃣ Normalize ratings (0–1)
      d.forEach(x=>x.rating_norm=(x.rating/10).toFixed(2));

```

```
      // 4 Simple TF-IDF token encoding
      let tfidf=new TFIDF();
      d.forEach(x=>tfidf.addDocument(x.review));
      d.forEach((x,i)=>x.vector=tfidf.listTerms(i).slice(0,5).map(v=>v.term).join(' '));

      // 5 Before vs after summary
      let rep={
        "Total Reviews":d.length,
        "Median Rating":med,
        "Avg Length":(d.reduce((a,b)=>a+b.review.length,0)/d.length).toFixed(1),
        "Example Encoded":d[0].vector
      };
      document.getElementById("report").textContent=JSON.stringify(rep,null,2);

      // 6 Download cleaned CSV
      const csv=Papa.unparse(d);
      const a=document.createElement("a");
      a.href=URL.createObjectURL(new Blob([csv],{type:"text/csv"}));
      a.download="cleaned_reviews.csv"; a.click();
    }
  });
}
</script>
</body>
</html>
```

OUTPUT:



CODE EXPLANATION:

This HTML and JavaScript code uploads a customer review CSV, cleans review text by converting to lowercase and removing HTML tags, fills missing ratings with the median, normalizes ratings to a 0–1 scale, encodes

reviews using a TF-IDF method, generates a before–after summary, and downloads the cleaned dataset.