CS 354 - Assignment 7

Arnav Jain - 220002018

Question 1

Code:

q1.cpp

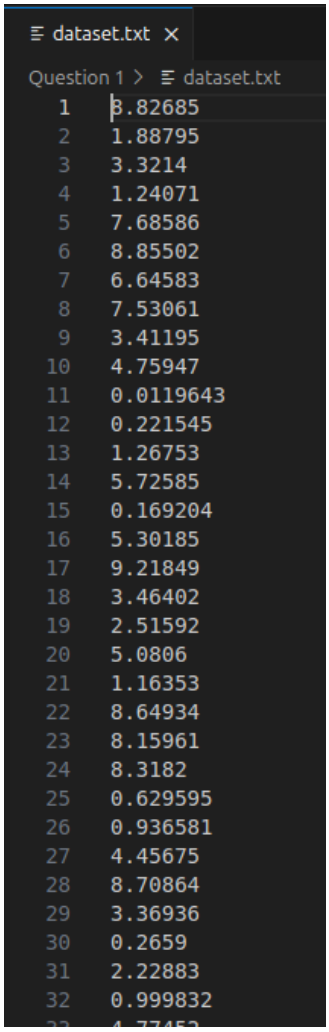
Question 1 > q1.cpp > SOM > SOM(int, float, float)

```cpp
#include <iostream>
#include <vector>
#include <random>
#include <cmath>
#include <fstream>

void save_dataset(const std::vector<float>& dataset, const std::string& filename) {
    std::ofstream out(filename);
    if(out) {
        for(const auto& val : dataset) {
            out << val << "\n";
        }
    }
}

class SOM {
private:
    std::vector<float> weights;
    float learning_rate;
    float radius;
    int num_neurons;

    float neighborhood(float dist, float sigma) {
        return exp(-(dist*dist)/(2*sigma*sigma));
    }

public:
    SOM(int n_neurons, float lr, float init_radius) :
        num_neurons(n_neurons),
        learning_rate(lr),
        radius(init_radius)
    {
        std::random_device rd;
        std::mt19937 gen(rd());
        std::uniform_real_distribution<float> dist(0.0f, 1.0f);

        weights.resize(n_neurons);
        for(auto& w : weights) {
            w = dist(gen);
        }
    }
```

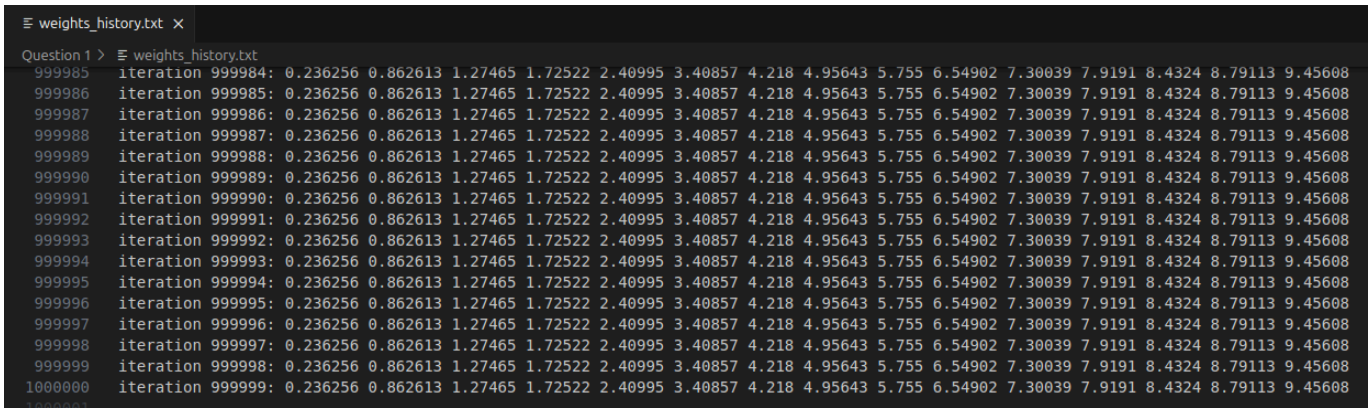
```
43      int find_bmu(float input) {
44          int bmu_idx = 0;
45          float min_dist = INFINITY;
46
47          for(int i=0; i<num_neurons; ++i) {
48              float dist = fabs(input - weights[i]);
49              if(dist < min_dist) {
50                  min_dist = dist;
51                  bmu_idx = i;
52              }
53          }
54          return bmu_idx;
55      }
56
57      void train(float input, int iteration, int max_iter) {
58          int bmu = find_bmu(input);
59
60          float current_lr = learning_rate * (1.0 - (float)iteration/max_iter);
61          float current_radius = radius * (1.0 - (float)iteration/max_iter);
62
63          for(int i=0; i<num_neurons; ++i) {
64              float distance_to_bmu = fabs(i - bmu);
65              float influence = neighborhood(distance_to_bmu, current_radius);
66              weights[i] += current_lr * influence * (input - weights[i]);
67          }
68      }
69
70      void save_weights(std::ofstream& out, int iteration) const {
71          out << "iteration " << iteration << ": ";
72          for(size_t i = 0; i < weights.size(); ++i) {
73              out << weights[i] << " ";
74          }
75          out << "\n";
76      }
77
78      const std::vector<float>& get_weights() const { return weights; }
79  };
```

```cpp
int main() {
    const int NUM_NEURONS = 15;
    const int EPOCHS = pow(10 , 6);
    const float INIT_LR = 0.01;
    const float INIT_RADIUS = NUM_NEURONS/2.0f;

    SOM som(NUM_NEURONS, INIT_LR, INIT_RADIUS);

    std::vector<float> dataset(100);
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_real_distribution<float> dist(0.0f, 10.0f);
    for(auto& val : dataset) {
        val = dist(gen);
    }

    save_dataset(dataset, "dataset.txt");

    std::ofstream weight_file("weights_history.txt");

    for(int epoch=0; epoch<EPOCHS; ++epoch) {
        int random_idx = std::rand() % dataset.size();
        float input = dataset[random_idx];

        som.train(input, epoch, EPOCHS);

        // Save weights for every iteration
        if(weight_file) {
            som.save_weights(weight_file, epoch);
        }
    }

    std::cout << "Training complete. Check weights_history.txt and dataset.txt\n";
    return 0;
}
```

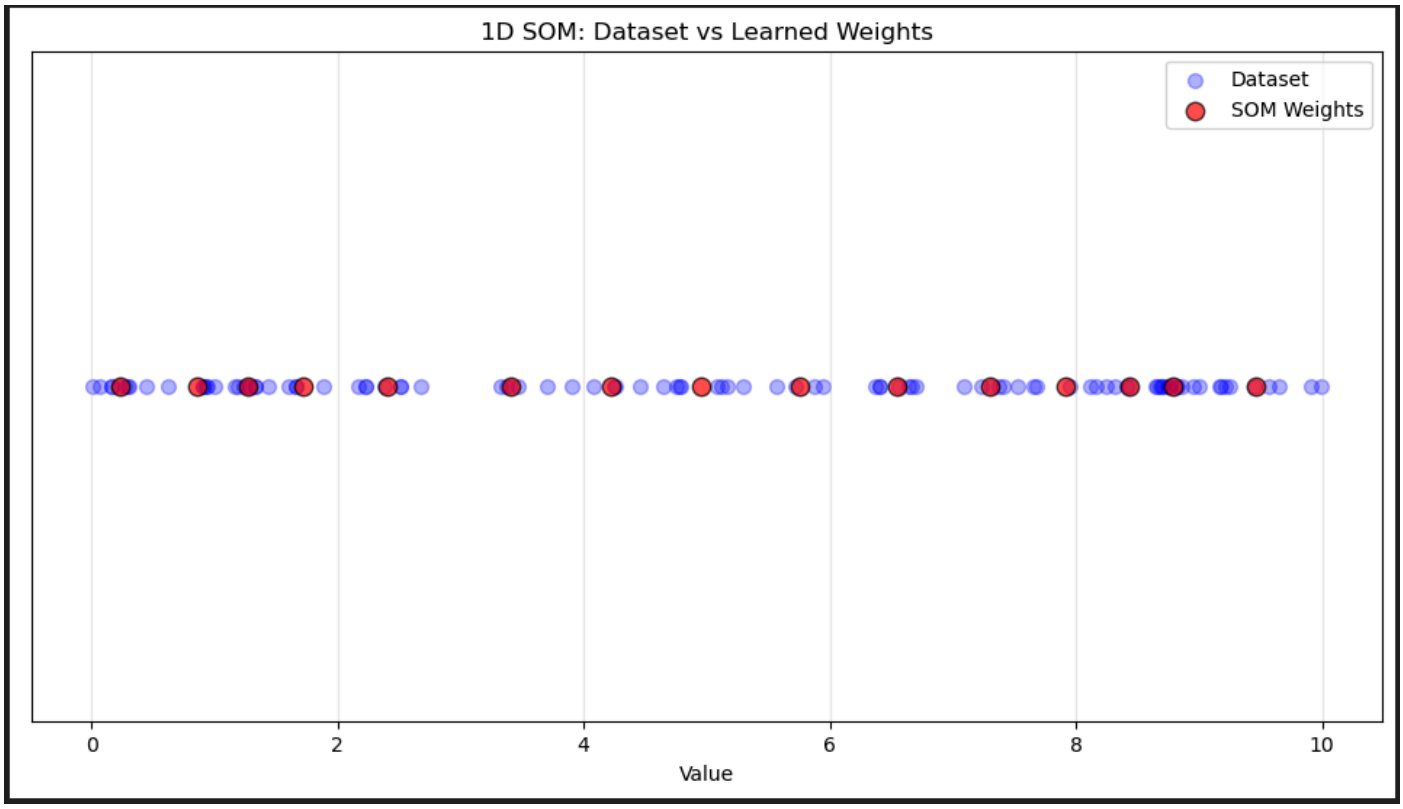
Dataset Generated:



Weights of SOM:



Visualisation of SOM:



Question 2

Code:

q2.cpp

Question 2 > q2.cpp > main()

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <sstream>
#include <cmath>
#include <random>
#include <algorithm>
#include <numeric>

using Matrix = std::vector<std::vector<double>>;
using namespace std;

Matrix load_csv(const std::string& filename) {
    Matrix data;
    std::ifstream file(filename);
    std::string line;

    // Skip header
    if(file.good()) std::getline(file, line);

    while(std::getline(file, line)) {
        std::vector<double> row;
        std::stringstream ss(line);
        std::string value;

        while(std::getline(ss, value, ',')) {
            try {
                row.push_back(std::stod(value));
            } catch(const std::exception& e) {
                std::cerr << "Error parsing value: " << value << "\n";
            }
        }
        if(!row.empty()) data.push_back(row);
    }
    return data;
}
```

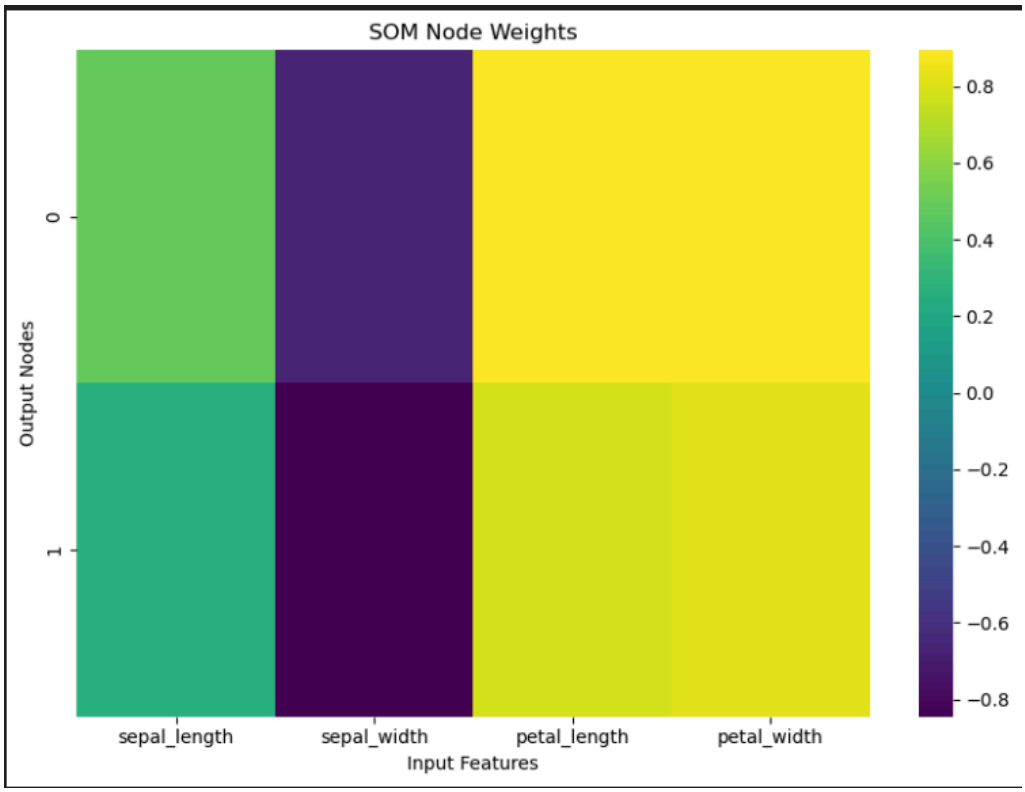
```cpp
void zscore_normalize(Matrix& data) {
    const size_t num_features = data[0].size();
    const size_t num_samples = data.size();

    for(size_t col=0; col<num_features; col++) {
        // Calculate mean
        double sum = std::accumulate(data.begin(), data.end(), 0.0,
            [col](double s, const std::vector<double>& row) { return s + row[col]; });
        double mean = sum / num_samples;

        // Calculate standard deviation
        double sq_diff = std::accumulate(data.begin(), data.end(), 0.0,
            [col, mean](double s, const std::vector<double>& row) {
                return s + pow(row[col] - mean, 2);
            });
        double std_dev = sqrt(sq_diff / num_samples);

        // Normalize
        for(auto& row : data) {
            row[col] = (row[col] - mean) / std_dev;
        }
    }
}

class KohonenSOM {
    Matrix weights;
    int input_size;
    int output_size;
    double initial_radius;

public:
    KohonenSOM(int n, int m) : input_size(n), output_size(m),
                               initial_radius(std::max(m, n)/2.0) {
        std::random_device rd;
        std::mt19937 gen(rd());
        std::uniform_real_distribution<double> dist(-1.0, 1.0);

        weights.resize(input_size, std::vector<double>(output_size));
        for(auto& row : weights) {
            std::generate(row.begin(), row.end(), [&](){ return dist(gen); });
        }
    }
```

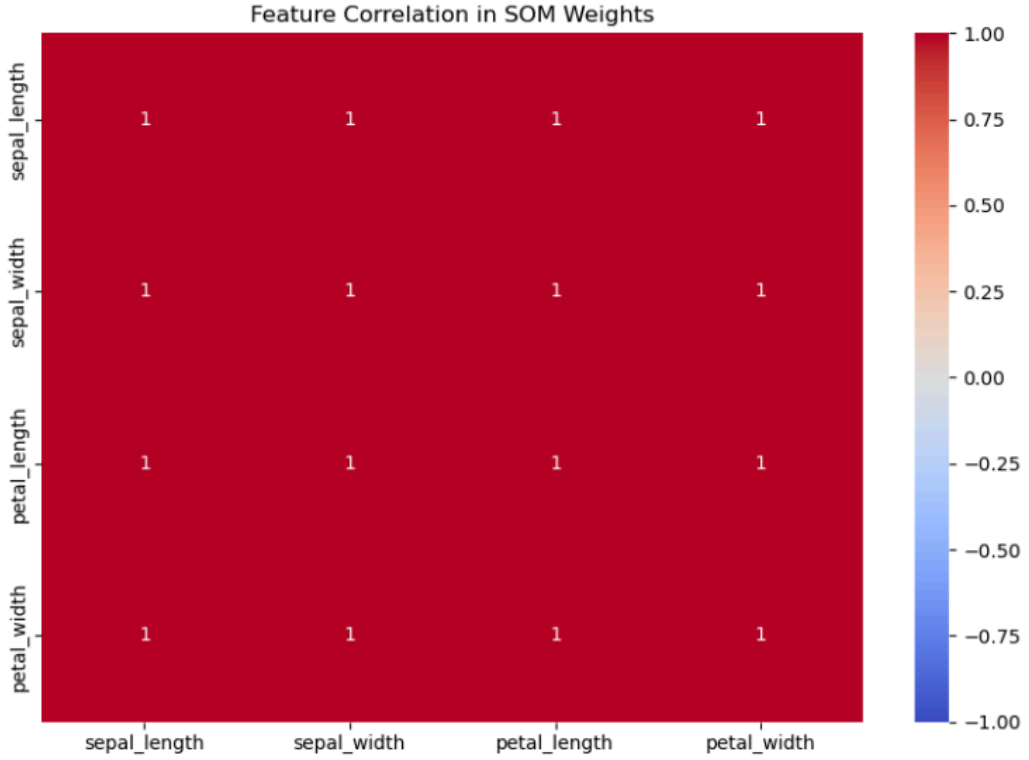
```
    int find_bmu(const std::vector<double>& input) {
        int bmu = 0;
        double min_dist = std::numeric_limits<double>::max();

        for(int j=0; j<output_size; j++) {
            double dist = 0.0;
            for(int i=0; i<input_size; i++) {
                dist += pow(input[i] - weights[i][j], 2);
            }
            if(dist < min_dist) {
                min_dist = dist;
                bmu = j;
            }
        }
        return bmu;
    }

    void train(const Matrix& data, int epochs) {
        const double initial_learning_rate = 0.5;
        const double time_constant = epochs / log(initial_radius);

        for(int epoch=0; epoch<epochs; epoch++) {
            double learning_rate = initial_learning_rate * exp(-epoch/static_cast<double>(epochs));
            double radius = initial_radius * exp(-epoch/time_constant);

            for(const auto& sample : data) {
                int bmu = find_bmu(sample);

                // Update neighborhood
                for(int j=0; j<output_size; j++) {
                    double distance_to_bmu = abs(j - bmu);
                    if(distance_to_bmu <= radius) {
                        double influence = exp(-pow(distance_to_bmu, 2)/(2*pow(radius, 2)));

                        for(int i=0; i<input_size; i++) {
                            weights[i][j] += learning_rate * influence *
                                             (sample[i] - weights[i][j]);
                        }
                    }
                }
            }
        }
    }
```

```cpp
    void save_weights(const std::string& filename) {
        std::ofstream outfile(filename);
        for(const auto& row : weights) {
            for(size_t i=0; i<row.size(); i++) {
                outfile << row[i] << (i < row.size()-1 ? "," : "");
            }
            outfile << "\n";
        }
    }
};

int main() {
    Matrix dataset = load_csv("iris_train.csv");
    zscore_normalize(dataset);

    const int num_features = dataset[0].size();
    cout<<"Sample Size: "<<dataset.size()<<endl;
    cout<<"Feature Size: "<<num_features<<endl;
    const int output_nodes = 2;

    KohonenSOM som(num_features, output_nodes);
    som.train(dataset, 50);
    som.save_weights("som_weights.csv");

    return 0;
}
```

SOM Weights:

Heatmaps:


SOM Node Weights
Output Nodes
0
1
sepal_length
sepal_width
petal_length
petal_width
Input Features
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8


Feature Correlation in SOM Weights
sepal_length
sepal_width
petal_length
petal_width
1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Question 3

```
from sklearn.datasets import load_iris
import pandas as pd

data = load_iris()
iris_df = pd.DataFrame(data.data, columns=data.feature_names)
iris_df['target'] = data.target
```

```
iris_df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
!pip install minisom
```

```
Collecting minisom
  Downloading minisom-2.3.5.tar.gz (12 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: minisom
  Building wheel for minisom (setup.py) ... done
  Created wheel for minisom: filename=MiniSom-2.3.5-py3-none-any.whl si
  Stored in directory: /root/.cache/pip/wheels/19/db/95/5e53bc2b88a3282
Successfully built minisom
Installing collected packages: minisom
Successfully installed minisom-2.3.5
```

```
iris = iris_df.iloc[: , :-1]
iris.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
from sklearn.preprocessing import MinMaxScaler
from minisom import MiniSom

# Normalization
scaler = MinMaxScaler()
normalized_data = scaler.fit_transform(iris_df.iloc[:, :-1])

# SOM initialization
som = MiniSom(x=10, y=10, input_len=4, sigma=1.0, learning_rate=0.5)
som.random_weights_init(normalized_data)
som.train_random(normalized_data, 100)
```

```
import numpy as np

def calculate_bmu_distances(som, data):
    return [np.linalg.norm(sample - som.get_weights()[som.winner(sample)])
            for sample in data]

distances = calculate_bmu_distances(som, normalized_data)
iris_df['bmu_distance'] = distances
```

```python
from scipy.stats import zscore

iris_df['z_score'] = zscore(distances)
anomalies = iris_df[iris_df['z_score'].abs() > 2]
```

```python
print(len(anomalies))
anomalies
```

```
7
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	bmu_distance	z_score
41	4.5	2.3	1.3	0.3	0	0.262963	2.631764
109	7.2	3.6	6.1	2.5	2	0.240869	2.302556
114	5.8	2.8	5.1	2.4	2	0.228376	2.116404
117	7.7	3.8	6.7	2.2	2	0.417237	4.930500
118	7.7	2.6	6.9	2.3	2	0.329537	3.623743
122	7.7	2.8	6.7	2.0	2	0.254917	2.511880
131	7.9	3.8	6.4	2.0	2	0.424832	5.043677

```python
import pandas as pd
import numpy as np

COLUMNS = [
    'duration', 'protocol_type', 'service', 'flag', 'src_bytes', 'dst_bytes',
    'land', 'wrong_fragment', 'urgent', 'hot', 'num_failed_logins',
    'logged_in', 'num_compromised', 'root_shell', 'su_attempted', 'num_root',
    'num_file_creations', 'num_shells', 'num_access_files', 'num_outbound_cmds',
    'is_host_login', 'is_guest_login', 'count', 'srv_count', 'serror_rate',
    'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
    'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count', 'dst_host_srv_count',
    'dst_host_same_srv_rate', 'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
    'dst_host_srv_diff_host_rate', 'dst_host_serror_rate', 'dst_host_srv_serror_rate',
    'dst_host_rerror_rate', 'dst_host_srv_rerror_rate',
]

def load_and_preprocess_kdd(file_path):
        df = pd.read_csv(
            file_path,
            header=None,
            names=COLUMNS,
            dtype={
                'protocol_type': 'category',
                'service': 'category',
                'flag': 'category',
            }
        )

        print("Data loaded successfully. First 5 rows:")
        print(df.head())

        categorical_cols = df.select_dtypes(include=['category']).columns

        print()
        print("Categorical column analysis:")
        for col in categorical_cols:
            unique_vals = df[col].unique()
            print(f"{col}: {len(unique_vals)} unique values")
            print(f"Sample values: {list(unique_vals[:3])}")
            print()
```

```
        # Vectorization strategy
        for col in categorical_cols:

            # One-hot encode
            if len(df[col].unique()) < 10:
                df = pd.concat([
                    df.drop(col, axis=1),
                    pd.get_dummies(df[col], prefix=col)
                ], axis=1)
            else:
                # Label encode
                df[col] = df[col].astype('category').cat.codes

        num_cols = df.select_dtypes(include=np.number).columns
        df[num_cols] = df[num_cols].astype('float32')

        return df
```

```
features = load_and_preprocess_kdd('/content/kddcup.testdata.unlabeled_10_percent')
```

```
Data loaded successfully. First 5 rows:
   duration protocol_type  service flag  src_bytes  dst_bytes  land  \
0         0           udp  private   SF        105        146     0
1         0           udp  private   SF        105        146     0
2         0           udp  private   SF        105        146     0
3         0           udp  private   SF        105        146     0
4         0           udp  private   SF        105        146     0

   wrong_fragment  urgent  hot  ...  dst_host_count  dst_host_srv_count  \
0               0       0    0  ...             255                 254
1               0       0    0  ...             255                 254
2               0       0    0  ...             255                 254
3               0       0    0  ...             255                 254
4               0       0    0  ...             255                 254

   dst_host_same_srv_rate  dst_host_diff_srv_rate  \
0                     1.0                    0.01
1                     1.0                    0.01
2                     1.0                    0.01
3                     1.0                    0.01
4                     1.0                    0.01
```

```
   dst_host_same_src_port_rate  dst_host_srv_diff_host_rate  \
0                         0.00                         0.0
1                         0.00                         0.0
...

flag: 11 unique values
Sample values: ['SF', 'RSTR', 'S1']
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```python
if features is not None:
    print()
    print("Processed features shape:", features.shape)
    print("Sample preprocessed data:")
    print(features.iloc[:3, :5])
```

```
Processed features shape: (311029, 43)
Sample preprocessed data:
   duration  service  flag  src_bytes  dst_bytes
0       0.0     12.0   5.0      105.0      146.0
1       0.0     12.0   5.0      105.0      146.0
2       0.0     12.0   5.0      105.0      146.0
```

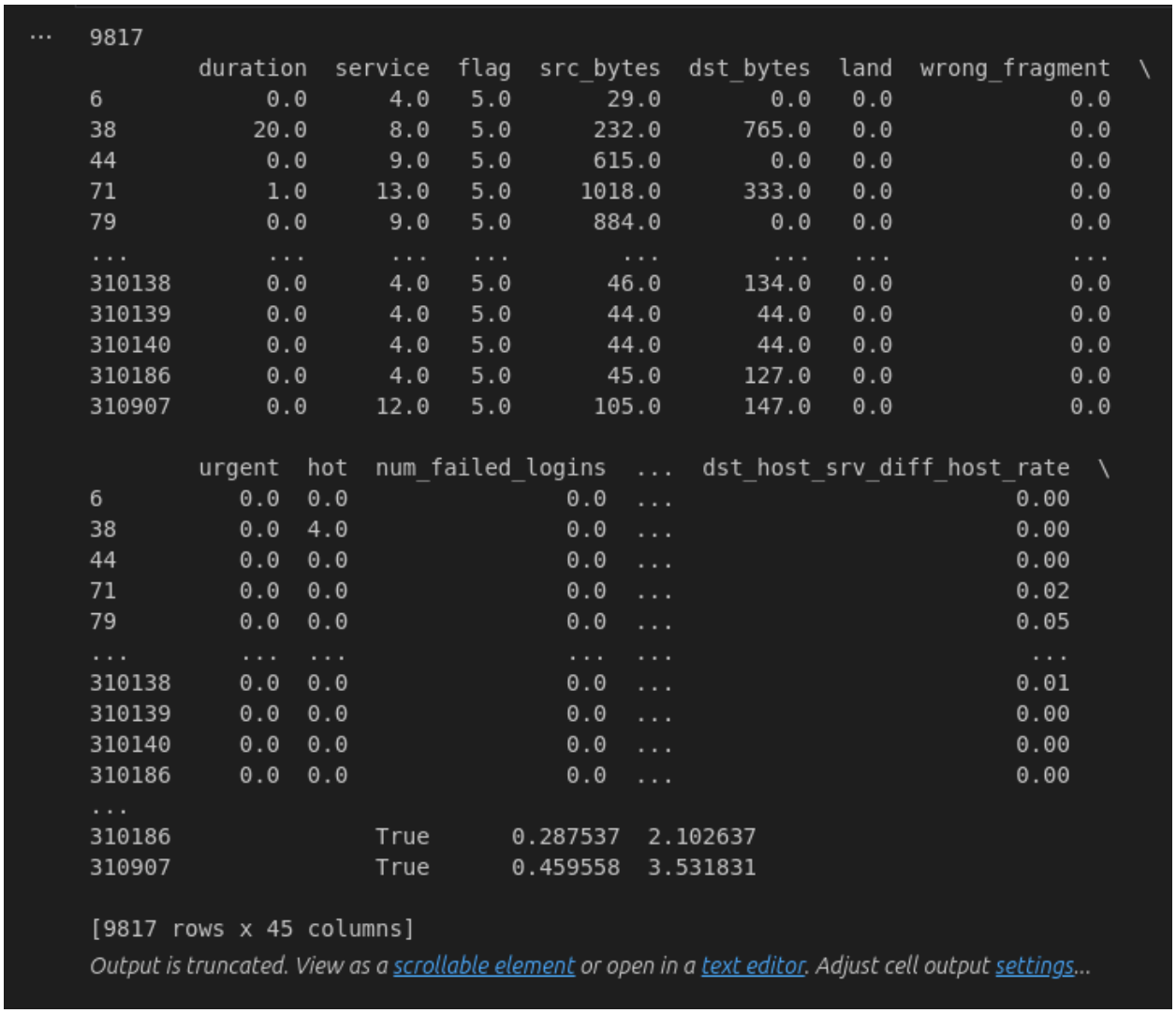
```python
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
som_input = scaler.fit_transform(features)
```

```python
# SOM initialization
som = MiniSom(x=50, y=50, input_len=43, sigma=1.0, learning_rate=0.5)
som.random_weights_init(som_input)
som.train_random(som_input, 1000)
```

```python
import numpy as np

def calculate_bmu_distances(som, data):
    return [np.linalg.norm(sample - som.get_weights()[som.winner(sample)])
            for sample in data]

distances = calculate_bmu_distances(som, som_input)
features['bmu_distance'] = distances
```

```python
from scipy.stats import zscore

features['z_score'] = zscore(distances)
anomalies = features[features['z_score'].abs() > 2]
```

```python
print(len(anomalies))
print(anomalies)
```



```
...   9817
             duration  service  flag  src_bytes  dst_bytes  land  wrong_fragment  \
      6           0.0      4.0   5.0       29.0        0.0   0.0             0.0
      38         20.0      8.0   5.0      232.0      765.0   0.0             0.0
      44          0.0      9.0   5.0      615.0        0.0   0.0             0.0
      71          1.0     13.0   5.0     1018.0      333.0   0.0             0.0
      79          0.0      9.0   5.0      884.0        0.0   0.0             0.0
      ...         ...      ...   ...        ...        ...   ...             ...
      310138      0.0      4.0   5.0       46.0      134.0   0.0             0.0
      310139      0.0      4.0   5.0       44.0       44.0   0.0             0.0
      310140      0.0      4.0   5.0       44.0       44.0   0.0             0.0
      310186      0.0      4.0   5.0       45.0      127.0   0.0             0.0
      310907      0.0     12.0   5.0      105.0      147.0   0.0             0.0

             urgent  hot  num_failed_logins  ...  dst_host_srv_diff_host_rate  \
      6         0.0  0.0                0.0  ...                         0.00
      38        0.0  4.0                0.0  ...                         0.00
      44        0.0  0.0                0.0  ...                         0.00
      71        0.0  0.0                0.0  ...                         0.02
      79        0.0  0.0                0.0  ...                         0.05
      ...       ...  ...                ...  ...                          ...
      310138    0.0  0.0                0.0  ...                         0.01
      310139    0.0  0.0                0.0  ...                         0.00
      310140    0.0  0.0                0.0  ...                         0.00
      310186    0.0  0.0                0.0  ...                         0.00
      ...
      310186              True      0.287537  2.102637
      310907              True      0.459558  3.531831

      [9817 rows x 45 columns]
      Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Github for code:

https://github.com/arnavjain2710/Computational-Intelligence-Lab-CS354N/tree/main/LAB%207