CSE 2794 – Machine Learning Workshop 2

LAB ASSIGNMENT-1: PREPROCESSING TECHNIQUES FOR NEURAL NETWORKS

Name: SK SHAKEEL AKHTAR

Reg. No.: 2341001063

Section: 23412C3

1. Data Exploration and Visualization

- Load the dataset and analyze its structure.
- Generate summary statistics and check for missing values.
- Visualize feature distributions (histograms, boxplots) to identify outliers.

```
import pandas as pd
import seaborn as sns

data = pd.read_csv('winequality-red.csv', delimiter=';')
data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
fixed acidity	1599.0	8.319637	1.741096	4.60000	7.1000	7.90000	9.200000	15.90000
volatile acidity	1599.0	0.527821	0.179060	0.12000	0.3900	0.52000	0.640000	1.58000
citric acid	1599.0	0.270976	0.194801	0.00000	0.0900	0.26000	0.420000	1.00000
residual sugar	1599.0	2.538806	1.409928	0.90000	1.9000	2.20000	2.600000	15.50000
chlorides	1599.0	0.087467	0.047065	0.01200	0.0700	0.07900	0.090000	0.61100
free sulfur dioxide	1599.0	15.874922	10.460157	1.00000	7.0000	14.00000	21.000000	72.00000
total sulfur dioxide	1599.0	46.467792	32.895324	6.00000	22.0000	38.00000	62.000000	289.00000
density	1599.0	0.996747	0.001887	0.99007	0.9956	0.99675	0.997835	1.00369
pH	1599.0	3.311113	0.154386	2.74000	3.2100	3.31000	3.400000	4.01000
sulphates	1599.0	0.658149	0.169507	0.33000	0.5500	0.62000	0.730000	2.00000
alcohol	1599.0	10.422983	1.065668	8.40000	9.5000	10.20000	11.100000	14.90000
quality	1599.0	5.636023	0.807569	3.00000	5.0000	6.00000	6.000000	8.00000

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

2. Data Cleaning

- Handle Missing Values (if present; else, explain potential methods).
- Outlier Treatment: Use IQR or Z-score to detect and handle outliers (e.g., capping).

```
def handle_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
# Capping outliers
    df[column] = df[column].clip(lower=lower_bound, upper=upper_bound)
    return df

# Apply IQR outlier treatment to all numerical columns
numerical_cols = data.select_dtypes(include=['number']).columns
for col in numerical_cols:
    data = handle_outliers_iqr(data, col)

# Display the updated data
print(data.describe().T)
```

```
                       count       mean        std       min      25%  \
fixed acidity         1599.0   8.290901   1.655860  4.600000   7.1000
volatile acidity      1599.0   0.526429   0.174045  0.120000   0.3900
citric acid           1599.0   0.270922   0.194614  0.000000   0.0900
residual sugar        1599.0   2.322358   0.609493  0.900000   1.9000
chlorides             1599.0   0.081194   0.017822  0.040000   0.0700
free sulfur dioxide   1599.0  15.689181   9.837494  1.000000   7.0000
total sulfur dioxide  1599.0  45.714822  30.374029  6.000000  22.0000
density               1599.0   0.996742   0.001806  0.992248   0.9956
pH                    1599.0   3.310353   0.149851  2.925000   3.2100
sulphates             1599.0   0.649831   0.137086  0.330000   0.5500
alcohol               1599.0  10.419627   1.054808  8.400000   9.5000
quality               1599.0   5.633521   0.783211  3.500000   5.0000

                           50%        75%         max
fixed acidity          7.90000   9.200000   12.350000
volatile acidity       0.52000   0.640000    1.015000
citric acid            0.26000   0.420000    0.915000
residual sugar         2.20000   2.600000    3.650000
chlorides              0.07900   0.090000    0.120000
free sulfur dioxide   14.00000  21.000000   42.000000
total sulfur dioxide  38.00000  62.000000  122.000000
density                0.99675   0.997835    1.001187
pH                     3.31000   3.400000    3.685000
sulphates              0.62000   0.730000    1.000000
alcohol               10.20000  11.100000   13.500000
quality                6.00000   6.000000    7.500000
```

3. Feature Engineering

- Create a Categorical Feature: Bin a numerical column (e.g., alcohol into low, medium, high).
- Encode Categorical Features: Apply one-hot encoding and label encoding.

```
# Define bins and labels for alcohol content
bins = [0, 9, 11, 100]  # Adjust these bins as needed
labels = ['low', 'medium', 'high']

# Create the categorical feature
data['alcohol_category'] = pd.cut(data['alcohol'], bins=bins, labels=labels, right=False)

# Display the updated data with the new feature
print(data.head())
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality alcohol_category
0      9.4      5.0           medium
1      9.8      5.0           medium
2      9.8      5.0           medium
3      9.8      6.0           medium
4      9.4      5.0           medium
```

```
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder, LabelEncoder

# Define bins and labels for alcohol content
bins = [0, 9, 11, 100]  # Adjust these bins as needed
labels = ['low', 'medium', 'high']

# Create the categorical feature
data['alcohol_category'] = pd.cut(data['alcohol'], bins=bins, labels=labels, right=False)

# One-hot encoding
encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False) # sparse=False for direct array access
encoded_data = encoder.fit_transform(data[['alcohol_category']])
encoded_df = pd.DataFrame(encoded_data, columns=encoder.get_feature_names_out(['alcohol_category']))
data = pd.concat([data, encoded_df], axis=1)

# Label encoding
label_encoder = LabelEncoder()
data['alcohol_category_label'] = label_encoder.fit_transform(data['alcohol_category'])

# Display the updated data with encoded features
print(data.head())
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality alcohol_category  alcohol_category_high  \
0      9.4      5.0           medium                    0.0
1      9.8      5.0           medium                    0.0
2      9.8      5.0           medium                    0.0
3      9.8      6.0           medium                    0.0
4      9.4      5.0           medium                    0.0

   alcohol_category_low  alcohol_category_medium  alcohol_category_label
0                   0.0                      1.0                       2
1                   0.0                      1.0                       2
2                   0.0                      1.0                       2
3                   0.0                      1.0                       2
4                   0.0                      1.0                       2
```

4. Feature Scaling

- Normalization: Scale features to [0, 1] using MinMaxScaler.
- Standardization: Use StandardScaler to transform data to mean=0, variance=1.

```
from sklearn.preprocessing import MinMaxScaler
# Assuming 'data' is your DataFrame from the previous code
# Create a MinMaxScaler object
scaler = MinMaxScaler()

# Select numerical features to scale (exclude the newly created categorical features)
numerical_cols_to_scale = ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'to

# Fit the scaler on the numerical features and transform them
data[numerical_cols_to_scale] = scaler.fit_transform(data[numerical_cols_to_scale])

# Display the updated data with scaled features
print(data.head())
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0       0.361290          0.648045     0.000000        0.363636     0.4500
1       0.412903          0.849162     0.000000        0.618182     0.7250
2       0.412903          0.715084     0.043716        0.509091     0.6500
3       0.851613          0.178771     0.612022        0.363636     0.4375
4       0.361290          0.648045     0.000000        0.363636     0.4500

   free sulfur dioxide  total sulfur dioxide   density        pH  sulphates  \
0             0.243902              0.241379  0.621085  0.769737   0.343284
1             0.585366              0.525862  0.509228  0.361842   0.522388
```

```
2            0.341463              0.413793  0.531600  0.440789   0.477612
3            0.390244              0.465517  0.643456  0.309211   0.373134
4            0.243902              0.241379  0.621085  0.769737   0.343284

    alcohol  quality alcohol_category  alcohol_category_high  \
0  0.196078      5.0           medium                    0.0
1  0.274510      5.0           medium                    0.0
2  0.274510      5.0           medium                    0.0
3  0.274510      6.0           medium                    0.0
4  0.196078      5.0           medium                    0.0

   alcohol_category_low  alcohol_category_medium  alcohol_category_label
0                   0.0                      1.0                       2
1                   0.0                      1.0                       2
2                   0.0                      1.0                       2
3                   0.0                      1.0                       2
4                   0.0                      1.0                       2
```

5. Feature Selection

- Principal Component Analysis (PCA): Apply PCA to reduce the number of features.

```
from sklearn.decomposition import PCA
import pandas as pd

# Assuming 'data' is your DataFrame from the previous code
# and numerical_cols_to_scale contains the numerical features

# Select numerical features for PCA
pca_features = data[numerical_cols_to_scale]

# Apply PCA with the desired number of components (e.g., 5 components)
pca = PCA(n_components=5)  # Adjust n_components as needed
pca_result = pca.fit_transform(pca_features)

# Create a DataFrame from the PCA results
pca_df = pd.DataFrame(data=pca_result, columns=[f'PC{i+1}' for i in range(5)])

# Concatenate the PCA results with the original DataFrame
data = pd.concat([data, pca_df], axis=1)

# Display the updated DataFrame
print(data.head())

# Explained variance ratio
explained_variance_ratio = pca.explained_variance_ratio_
print("Explained Variance Ratio:", explained_variance_ratio)
print("Total Variance Explained:", sum(explained_variance_ratio))
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0       0.361290          0.648045     0.000000        0.363636     0.4500
1       0.412903          0.849162     0.000000        0.618182     0.7250
2       0.412903          0.715084     0.043716        0.509091     0.6500
3       0.851613          0.178771     0.612022        0.363636     0.4375
4       0.361290          0.648045     0.000000        0.363636     0.4500

   free sulfur dioxide  total sulfur dioxide   density        pH  sulphates  \
0             0.243902              0.241379  0.621085  0.769737   0.343284
1             0.585366              0.525862  0.509228  0.361842   0.522388
2             0.341463              0.413793  0.531600  0.440789   0.477612
3             0.390244              0.465517  0.643456  0.309211   0.373134
4             0.243902              0.241379  0.621085  0.769737   0.343284

   ...  alcohol_category  alcohol_category_high alcohol_category_low  \
0  ...            medium                    0.0                  0.0
1  ...            medium                    0.0                  0.0
2  ...            medium                    0.0                  0.0
3  ...            medium                    0.0                  0.0
4  ...            medium                    0.0                  0.0

   alcohol_category_medium  alcohol_category_label       PC1       PC2  \
0                      1.0                       2 -0.334066 -0.044274
1                      1.0                       2 -0.081086  0.415474
2                      1.0                       2 -0.103170  0.148488
3                      1.0                       2  0.454986  0.010979
4                      1.0                       2 -0.334066 -0.044274

         PC3       PC4       PC5
0 -0.379685 -0.078170 -0.012338
1 -0.311652  0.062130  0.126833
2 -0.313668 -0.015201  0.087897
3  0.178805 -0.326279 -0.266847
4 -0.379685 -0.078170 -0.012338
```

```
[5 rows x 22 columns]
Explained Variance Ratio: [0.26734621 0.21923543 0.14619164 0.09807839 0.07608929]
Total Variance Explained: 0.8069409592967512
```

6. Train Test Splitting

- Split the dataset so that 80% of the samples will be used for training and rest will be used for testing.

```
from sklearn.model_selection import train_test_split

# Assuming 'data' is your DataFrame and 'quality' is your target variable
X = data.drop('quality', axis=1)
y = data['quality']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # random_state for reproducibility

# Now you have X_train, X_test, y_train, and y_test
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(1279, 21) (320, 21) (1279,) (320,)
```