MTMC-512 Programming Lab IV(Machine Learning)

Lab Assignment 2: Data Preprocessing Pipeline

1 Task 1: Load and Explore the Dataset

1.1 Loading the Dataset

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('heart_disease_uci.csv')
df.head()
```

Output:

```
   id  age     sex   dataset               cp  trestbps   chol    fbs \
0   1   63    Male  Cleveland   typical angina     145.0  233.0   True
1   2   67    Male  Cleveland     asymptomatic     160.0  286.0  False
2   3   67    Male  Cleveland     asymptomatic     120.0  229.0  False
3   4   37    Male  Cleveland      non-anginal     130.0  250.0  False
4   5   41  Female  Cleveland  atypical angina     130.0  204.0  False
```

1.2 Dataset Characteristics

```
df.shape
df.dtypes
df.describe()
df.isnull().sum()
df.duplicated().sum()
```

Output:

```
Shape: (920, 16)

Data Types:
id              int64
age             int64
sex            object
dataset        object
cp             object
trestbps      float64
chol          float64
fbs            object
restecg        object
thalch        float64
exang          object
oldpeak       float64
slope          object
```

```
ca           float64
thal          object
num            int64

Missing Values:
trestbps     59
chol         30
fbs          90
restecg       2
thalch       55
exang        55
oldpeak      62
slope       309
ca          611
thal        486
num           0
```

2 Task 2: Data Cleaning

2.1 Handling Missing Values

```
numeric = df.select_dtypes(include=['float64','int64']).columns
df[numeric] = df[numeric].fillna(df[numeric].median())

categoric = df.select_dtypes(include=['object']).columns
df[categoric] = df[categoric].fillna(df[categoric].mode())
```

Output:

```
Missing values after imputation:
id            0
age           0
sex           0
dataset       0
cp            0
trestbps      0
chol          0
fbs           0
restecg       0
thalch        0
exang         0
oldpeak       0
slope         0
ca            0
thal          0
num           0
```

2.2 Boxplot for Visualizing Outliers

```
num_columns = df.select_dtypes(include=['float64', 'int64']).columns
plt.figure(figsize = (20,12))
sns.boxplot(data=df[num_columns])
plt.show()
```

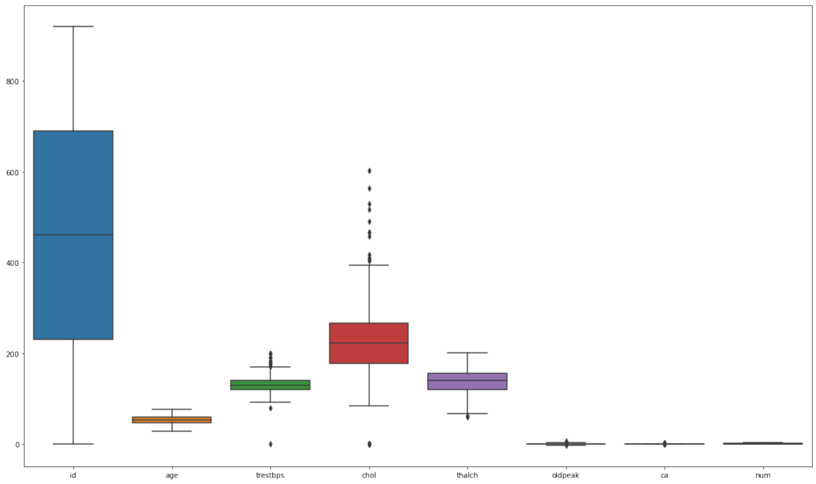

Figure 1: Box Plot For Outliers

2.3 Handling Outliers Using Z-scores

```
from scipy.stats import zscore
z_scores = df[numeric].apply(zscore)

zero_outliers = df[(z_scores < 3).all(axis=1)]
df = zero_outliers

df_shape = df.shape
```

Output:

```
Data after removing outliers using Z-scores: (887, 16)
```

3 Task 3: Feature Engineering

3.1 Encoding Categorical Features

```
df = pd.get_dummies(df, columns=categoric, drop_first=True)
```

3.2 Feature Scaling

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[numeric] = scaler.fit_transform(df[numeric])
```

3.3 Correlation Analysis

```
corr = df.corr()
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.savefig("heatmap.png")
plt.show()
```

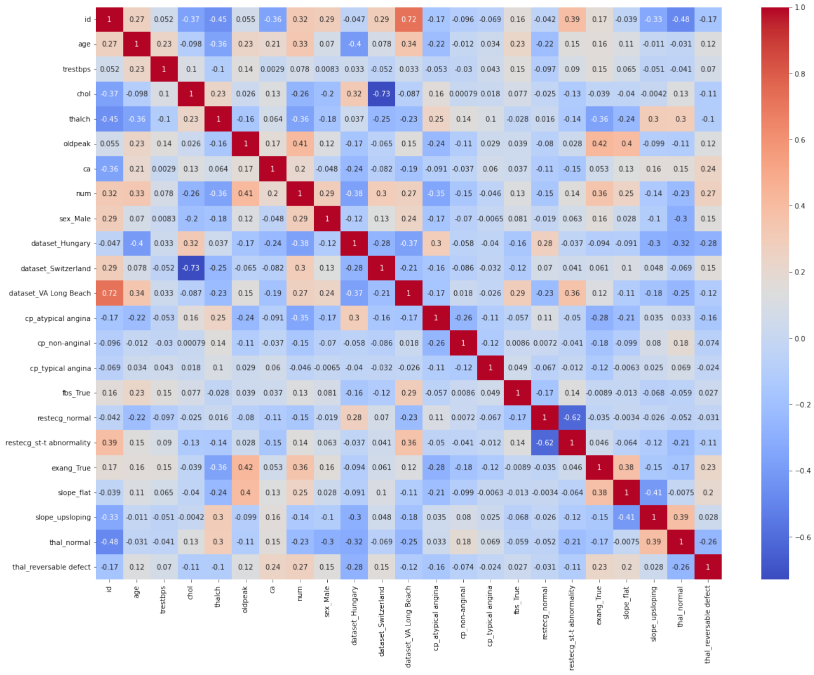

Figure 2: Feature Correlation Heatmap

4 Task 4: Classification using kNN

4.1 Splitting the Dataset

```
from sklearn.model_selection import train_test_split
X = df.drop('num',axis=1)
y = df['num']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)
```

4.2 Training k-Nearest Neighbors (kNN)

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
```

4.3 Model Evaluation

```
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

Output:

```
Accuracy: 0.8595505617977528

Classification Report:
```

```
              precision    recall  f1-score   support
           0       0.88      0.97      0.92       155
           1       0.33      0.10      0.15        20
           2       0.00      0.00      0.00         3
    accuracy                           0.86       178
   macro avg       0.40      0.36      0.36       178
weighted avg       0.80      0.86      0.82       178

Confusion Matrix:
[[151   4   0]
 [ 18   2   0]
 [  3   0   0]]
```

5 Conclusion

In conclusion, while the current model demonstrates a solid classification performance, there's room for improvement, particularly by experimenting with more advanced algorithms and fine-tuning the current model. Preprocessing steps, including outlier handling, missing value imputation, and feature scaling, were crucial in ensuring the dataset was ready for training and ultimately contributed to the overall model accuracy.