Heart Disease Data Analysis Report

Step 1: Data Loading

Objective: Load the dataset from a CSV file into a pandas DataFrame.

Code:

```
data_path = r'C:\Users\salma\PycharmProjects\assignment-2\heart_disease_uci.csv'

data = pd.read_csv(data_path)
```

Explanation:

This code loads the dataset from the specified path using pandas' read_csv function, storing the data in a DataFrame called data.

Heart Disease Data Analysis Report

Step 2: Initial Data Summary

Objective: Display basic statistical summaries and information about the dataset.

Code:

```
print(data.describe(include='all'))

print(data.info())
```

Explanation:

data.describe(include='all') provides a statistical summary for all columns, including count, mean, std, min, max, and frequency for categorical data. data.info() shows a concise summary of the DataFrame, including the number of non-null entries and the data type of each column.

Heart Disease Data Analysis Report

Step 3: Missing Values Handling

Objective: Impute missing values for both numerical and categorical data.

Code:

```
def safe_impute(column, strategy='median'):
    imputer = SimpleImputer(strategy=strategy)
    transformed = imputer.fit_transform(data[[column]])
    return pd.Series(transformed.ravel(), index=data.index)


data['trestbps'] = safe_impute('trestbps', 'median')
data['chol'] = safe_impute('chol', 'median')
data['fbs'] = safe_impute('fbs', 'most_frequent')
data['restecg'] = safe_impute('restecg', 'most_frequent')
data['slope'] = safe_impute('slope', 'most_frequent')
```

Explanation:

Missing values in numerical columns ('trestbps', 'chol') are imputed using the median of the column, and categorical columns ('fbs', 'restecg', 'slope') use the mode. This ensures that the data does not contain null values, which can affect analysis and model training.

Heart Disease Data Analysis Report

Step 4: Outliers Handling

Objective: Cap outliers in the 'trestbps' and 'chol' columns at the 95th percentile to minimize their effect.

Code:

```
data['trestbps'] = np.where(data['trestbps'] > data['trestbps'].quantile(0.95), data['trestbps'].quantile(0.95), data['trestbps'])
data['chol'] = np.where(data['chol'] > data['chol'].quantile(0.95), data['chol'].quantile(0.95), data['chol'])
```

Explanation:

This code limits the maximum value of 'trestbps' and 'chol' to their respective 95th percentile values. It effectively reduces the range of extreme values which can skew the analysis.

Step 5: Feature Engineering

Objective: Create new features to potentially enhance model performance by capturing interactions and non-linear relationships.

Code:

```
data['age_thalch_interaction'] = data['age'] * data['thalch']
data['trestbps_chol_interaction'] = data['trestbps'] * data['chol']
data['age_squared'] = data['age'] ** 2
age_bins = pd.cut(data['age'], bins=[0, 30, 45, 60, 75, 100], labels=['Under 30', '31-45', '46-60', '61-75', 'Over 75'])
data['age_bins'] = age_bins
```

Explanation:

Interactions between 'age' and 'thalch', and between 'trestbps' and 'chol', are created to explore possible combined effects on heart disease. Age is also squared to capture its non-linear impact, and binned into categories for better grouping in analysis.