

Practical Machine Learning and Data Exploration
First Trimester 2025

Assignment 2

Heart Disease Dataset

Python Code File

ID : 243002097

Name : Ahmad Mostafa Abutalib

Introduction:

In this project, we performed data cleaning by handling missing values to prepare the dataset for machine learning applications.

Although not explicitly required, outlier values were also detected and treated to enhance model performance and ensure data consistency.

Feature selection was conducted using two different methods to identify the most important attributes, and machine learning models were applied with Grid Search optimization to achieve the best evaluation results.

Step 1: Import Required Libraries

```
import pandas as pd
```

```
import numpy as np
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.model_selection import train_test_split, GridSearchCV
```

```
from sklearn.feature_selection import SelectKBest, chi2
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.metrics import classification_report, accuracy_score
```

Step 2: Load Dataset

```
df = pd.read_csv('heart.csv')
```

Step 3: Handle Missing Values

```
for column in df.columns:
```

```
    if df[column].dtype == 'object':
```

```
        df[column].fillna(df[column].mode()[0], inplace=True)
```

```
    else:
```

```
        df[column].fillna(df[column].mean(), inplace=True)
```

Step 4: Handle Outliers using IQR

```
for col in df.select_dtypes(include=[np.number]).columns:
```

```
    if col == 'target':
```

```
        continue
```

```
    Q1 = df[col].quantile(0.25)
```

```
    Q3 = df[col].quantile(0.75)
```

```
    IQR = Q3 - Q1
```

```
    lower = Q1 - 1.5 * IQR
```

```
    upper = Q3 + 1.5 * IQR
```

```
    df[col] = np.where((df[col] < lower) | (df[col] > upper), np.nan, df[col])
```

Fill NaNs after outlier removal

```
for column in df.columns:
```

```
    if df[column].dtype != 'object':
```

```
        df[column].fillna(df[column].mean(), inplace=True)
```

Step 5: Feature Selection

```
X = df.drop('target', axis=1)
```

```
y = df['target']
```

Method 1: SelectKBest

```
selector = SelectKBest(score_func=chi2, k=5)

X_new = selector.fit_transform(X, y)

selected_features_kbest = X.columns[selector.get_support()]

print("Top features from SelectKBest:", selected_features_kbest.tolist())
```

Method 2: Random Forest Feature Importance

```
rf = RandomForestClassifier()

rf.fit(X, y)

importances = pd.Series(rf.feature_importances_, index=X.columns)

important_features_rf = importances.nlargest(5).index

print("Top features from RandomForest:", important_features_rf.tolist())
```

Step 6: Modeling with GridSearchCV

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Using Decision Tree with GridSearchCV

```
param_grid = {'max_depth': [3, 5, 7, None]}

grid = GridSearchCV(DecisionTreeClassifier(), param_grid, cv=5)

grid.fit(X_train, y_train)

best_model = grid.best_estimator_
```



```
y_pred = best_model.predict(X_test)
```

```
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
# Step 7: Save Cleaned Dataset
```

```
df.to_csv('Ahmad_AbuTalib_Heart_Disease_Cleaned_Full.csv', index=False)
```