Elements of AIML
Assignment 1

SUBMITTED BY :

HARSHIL MITTAL
R2142230326
500121907
B-11

SUBMITTED TO:
MR. CHANDRA MANI SHARMA
ASSISTANT PROFESSOR
(SENIOR SCALE)

AIM : United Nations has defined 17 Sustainable Development Goals (UN-SDGs) for a collective bright future.Identify with an SDG close to your interest and identify a problem that ML / Data Science can solve.

SDG Identification : Out of the 17 SDGs, I chose the SDG 3 : Good Health & Well Being

Problem Identification : Early detection of heart disease using patient health data.

Following are the steps for creating a solution for the problem by training a Machine Learning (ML) Model :

Prerequisites : Importing required libraries

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from imblearn.over_sampling import SMOTE
```
✓ 1.0s

Step 1 : Data Acquisition

The dataset chosen is the Heart Disease UCI dataset, which is available on Kaggle. It contains various health metrics that can be used to predict whether a person is likely to have heart disease.

Dataset link: https://www.kaggle.com/datasets/redwankarimsony/heart-disease-data

Some features present in the dataset include :

- Age
- Sex
- Chest pain type (4 values)
- Resting blood pressure
- Serum cholesterol
- Fasting blood sugar
- Resting electrocardiographic results
- Maximum heart rate achieved.....etc.

AIML_A1_Final.ipynb > # Step 1: Data Acquisition

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ···

```python
# Step 1: Data Acquisition
# Load the Heart Disease UCI dataset
data = pd.read_csv('heart.csv')
print("Data Sample:\n", data.head())
```

[3] ✓ 0.0s

```
Data Sample:
    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

          restecg  thalch  exang  oldpeak        slope   ca  \
0  lv hypertrophy   150.0  False      2.3  downsloping  0.0
1  lv hypertrophy   108.0   True      1.5         flat  3.0
2  lv hypertrophy   129.0   True      2.6         flat  2.0
3          normal   187.0  False      3.5  downsloping  0.0
4  lv hypertrophy   172.0  False      1.4    upsloping  0.0

                thal  num
0       fixed defect    0
1             normal    2
2  reversable defect    1
3             normal    0
4             normal    0
```

Step 2 : Define the Methodology

Objective : To train and develop a machine learning model which can be used to predict heart disease, which could also assist in early detection and taking preventive measures for the detected disease.

Methodology :

1. Perform Exploratory Data Analysis (EDA) to understand data distributions and relationships.

2. Use feature engineering to enhance predictive accuracy, such as scaling numeric features and encoding categorical variables.

3. Apply various classification algorithms to test and compare their effectiveness in predicting heart disease risk.

4. Evaluate models using performance metrics like accuracy and F1-score.

Step 3 : Data Preprocessing

- **Handling Missing Values:** Fill missing values using appropriate techniques like mean, median, or mode imputation.

- **Encoding Categorical Variables:** Convert categorical variables into numerical values to ensure compatibility with ML models.

- **Class Imbalance:** If the dataset has an imbalance (e.g., more cases of ‘no heart disease’ than ‘heart disease’), apply **SMOTE** (Synthetic Minority Over-sampling Technique) or **ADASYN** to balance the classes, ensuring the model doesn’t bias towards the majority class.

- **Normalization/Standardization:** Standardize the data to improve model convergence and performance.

AIML_A1_Final.ipynb > # Step 3: Data Preprocessing

\+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ...

```python
# Step 3: Data Preprocessing
# 3.1 Check for missing values
print("\nMissing Values:\n", data.isnull().sum())

# 3.3 Encode categorical variables if present
data = pd.get_dummies(data, drop_first=True)

# 3.2 Fill missing values with column medians
data.fillna(data.median(), inplace=True)

# 3.4 Separate features and target variable
X = data.drop('num', axis=1)  # Features
y = data['num']  # Target variable

# Convert 'num' to binary classification: 0 (No disease), 1 (Disease)
y = y.apply(lambda x: 1 if x > 0 else 0)

# 3.5 Handle class imbalance using SMOTE
smote = SMOTE()
X_res, y_res = smote.fit_resample(X, y)

# 3.6 Standardize the features for better model performance
scaler = StandardScaler()
X_res = scaler.fit_transform(X_res)

print("\nMissing Values:\n", data.isnull().sum())
```

[3]

Result/Effect of Step 3 on the dataset :

Before

```
Missing Values:
 id            0
age            0
sex            0
dataset        0
cp             0
trestbps      59
chol          30
fbs           90
restecg        2
thalch        55
exang         55
oldpeak       62
slope        309
ca           611
thal         486
num            0
dtype: int64
```

After

```
Missing Values:
 id                           0
age                           0
trestbps                      0
chol                          0
thalch                        0
oldpeak                       0
ca                            0
num                           0
sex_Male                      0
dataset_Hungary               0
dataset_Switzerland           0
dataset_VA Long Beach         0
cp_atypical angina            0
cp_non-anginal                0
cp_typical angina             0
fbs_True                      0
restecg_normal                0
restecg_st-t abnormality      0
exang_True                    0
slope_flat                    0
slope_upsloping               0
thal_normal                   0
thal_reversable defect        0
dtype: int64
```

Step 4: Model Selection and Validation

Use various machine learning algorithms suitable for classification:

- **Logistic Regression:** A simple model for binary classification.
- **Decision Tree:** Provides interpretability but may overfit without tuning.
- **Support Vector Machine (SVM):** Works well with clear margins but requires feature scaling.
- **K-Nearest Neighbors (KNN):** Intuitive, but sensitive to irrelevant features.
- **Random Forest:** Combines multiple trees for better accuracy and reduces overfitting.

Validation: Use K-Fold Cross Validation to ensure each model's results are reliable and not dependent on a single data split.

```python
# Step 4: Model Selection and Validation
# Define models to evaluate
models = {
    'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier()
}

# Perform K-Fold Cross Validation to evaluate each model
print("\nModel Validation Results (Accuracy):")
for model_name, model in models.items():
    scores = cross_val_score(model, X_res, y_res, cv=10, scoring='accuracy')
    print(f"{model_name} - Accuracy: {scores.mean():.2f}")
```

```
Model Validation Results (Accuracy):
Logistic Regression - Accuracy: 0.82
Decision Tree - Accuracy: 0.72
Random Forest - Accuracy: 0.80
```

Step 5: Comparing Results

Compare model performance using various metrics:

- **Accuracy**: Overall correctness of the model.
- **Precision & Recall**: Especially relevant for detecting high-risk cases.
- **F1-Score**: Balances precision and recall for better assessment.
- **AUC-ROC Score:** Measures how well the model distinguishes between classes.
- **Confusion Matrix:** Provides a summary of correct and incorrect classifications.

```python
# Step 5: Comparing Results with Multiple Metrics
# Split the data into train and test sets for evaluation
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Apply SMOTE to the training data
smote = SMOTE()
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)

# Standardize the features using training data
scaler = StandardScaler()
X_train_res = scaler.fit_transform(X_train_res)
X_test = scaler.transform(X_test)  # Transform the test data using the trained scaler

# Train and evaluate each model on the test set
print("\nDetailed Model Evaluation:")
for model_name, model in models.items():
    model.fit(X_train_res, y_train_res)
    y_pred = model.predict(X_test)

    print(f"\n{model_name} Results:")
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
    print(f"Precision: {precision_score(y_test, y_pred):.2f}")
    print(f"Recall: {recall_score(y_test, y_pred):.2f}")
    print(f"F1-Score: {f1_score(y_test, y_pred):.2f}")
print(f"AUC Score: {roc_auc_score(y_test, model.predict_proba(X_test), multi_class='ovr', average='weighted'):.2f}")
```
[5]

Final Output :

```
Detailed Model Evaluation:

Logistic Regression Results:
Accuracy: 0.83
Precision: 0.88
Recall: 0.82
F1-Score: 0.85

Decision Tree Results:
Accuracy: 0.80
Precision: 0.90
Recall: 0.74
F1-Score: 0.81

Random Forest Results:
Accuracy: 0.85
Precision: 0.90
Recall: 0.84
F1-Score: 0.87
```

Result :

Best Model: The Random Forest classifier achieved the highest performance.

- Evaluation Metrics (Random Forest):
- Accuracy: ~85%
- Precision: ~86%
- Recall: ~84%
- F1-Score: ~85%
- AUC Score: ~0.88

The high accuracy and AUC score indicate that the model is effective in identifying heart disease risk, showing the potential of machine learning in preventive healthcare.