Heart Disease Prediction using SVM

.

Summary

This project performs data preprocessing, exploratory data analysis (EDA), model training using Support Vector Machine (SVM), evaluation, and model saving. Below you will find a concise report, the original code from your notebook, and key outputs (metrics and EDA plots).

Introduction:

The goal is to predict heart disease using clinical features. The workflow includes cleaning, EDA, SVM training, evaluation, and model saving.

Key Result:

Test accuracy: approx. 77% (see model metrics file).

Sample EDA Figures:

.

Project Code :

```
import pandas as pd
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
import pickle

class ModelTraining:
    def __init__(self, X_train, X_test, y_train, y_test):
        self.X_train = X_train
        self.X_test = X_test
        self.y_train = y_train
        self.y_test = y_test
        self.model = None

    def train_model(self, kernel='linear'):
        """Train SVM model"""
        self.model = SVC(kernel=kernel, random_state=42)
        self.model.fit(self.X_train, self.y_train)
        print("\n Model training completed.")
        return self.model
```

```
    def evaluate_model(self):
        """Evaluate model on test set"""
        y_pred = self.model.predict(self.X_test)

        acc = accuracy_score(self.y_test, y_pred)
        report = classification_report(self.y_test, y_pred)
        cm = confusion_matrix(self.y_test, y_pred)

        print(f"\n Accuracy: {acc:.4f}")
        print("\n Classification Report:")
        print(report)
        print("\n Confusion Matrix:")
        print(cm)

        return acc, report, cm


if __name__ == "__main__":
    df = pd.read_csv("cleaned_heart_.csv")

    X = df.drop(columns=['target'])
    y = df['target']

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

    trainer = ModelTraining(X_train, X_test, y_train, y_test)
    trainer.train_model(kernel='linear')
    trainer.evaluate_model()

    with open("heart_svm.pkl", "wb") as f:
        pickle.dump(trainer.model, f)
    print("\n Model saved successfully as heart_svm.pkl")


# ---- Next cell ----

import pickle

class ModelPersistence:
    def __init__(self, model):
        self.model = model

    def save_model(self, filename="svm_model.pkl"):
        """Save trained model to a pickle file"""
        with open(filename, "wb") as file:
            pickle.dump(self.model, file)
        print(f"\n Model saved successfully as {filename}")

    @staticmethod
    def load_model(filename="svm_model.pkl"):
        """Load model from pickle file"""
        with open(filename, "rb") as file:
            model = pickle.load(file)
        print(f"\n Model loaded successfully from {filename}")
        return model


if __name__ == "__main__":
    import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.svm import SVC

    df = pd.read_csv(r"cleaned_heart_.csv")
    X = df.drop(columns=['target'])
    y = df['target']

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

    model = SVC(kernel='linear', random_state=42)
    model.fit(X_train, y_train)

    saver = ModelPersistence(model)
    saver.save_model("heart_svm.pkl")
    loaded_model = ModelPersistence.load_model("heart_svm.pkl")
```

```
sample_pred = loaded_model.predict(X_test[:5])
print("\n Sample Predictions on first 5 rows of Test Set:")
print(sample_pred)
```