





import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix, roc\_curve, auc, roc\_auc\_score

from sklearn.preprocessing import label\_binarize

from sklearn.multiclass import OneVsRestClassifier

from itertools import cycle

# Load the breast cancer dataset

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data"

column\_names = ['ID', 'Diagnosis', 'Mean Radius', 'Mean Texture', 'Mean Perimeter', 'Mean Area', 'Mean Smoothness',

'Mean Compactness', 'Mean Concavity', 'Mean Concave Points', 'Mean Symmetry', 'Mean Fractal Dimension',

'SE Radius', 'SE Texture', 'SE Perimeter', 'SE Area', 'SE Smoothness', 'SE Compactness', 'SE Concavity',

'SE Concave Points', 'SE Symmetry', 'SE Fractal Dimension', 'Worst Radius', 'Worst Texture',

'Worst Perimeter', 'Worst Area', 'Worst Smoothness', 'Worst Compactness', 'Worst Concavity',

'Worst Concave Points', 'Worst Symmetry', 'Worst Fractal Dimension']

data = pd.read\_csv(url, header=None, names=column\_names)

# Drop unnecessary columns (e.g., ID column)

data = data.drop(['ID'], axis=1)

# Convert categorical diagnosis labels to numerical (Malignant: 1, Benign: 0)

data['Diagnosis'] = data['Diagnosis'].map({'M': 1, 'B': 0})

# Separate features (X) and target variable (y)

X = data.drop('Diagnosis', axis=1)

y = data['Diagnosis']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize the Random Forest classifier

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the classifier on the training data

rf\_classifier.fit(X\_train, y\_train)

# Feature Importance Plot

feature\_importance = rf\_classifier.feature\_importances\_

feature\_names = X.columns

sorted\_idx = np.argsort(feature\_importance)

plt.figure(figsize=(10, 8))

plt.barh(range(len(sorted\_idx)), feature\_importance[sorted\_idx], align="center")

plt.yticks(range(len(sorted\_idx)), feature\_names[sorted\_idx])

plt.xlabel("Feature Importance")

plt.title("Random Forest Feature Importance")

plt.show()

# Confusion Matrix Plot

y\_pred = rf\_classifier.predict(X\_test)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(6, 6))

plt.imshow(conf\_matrix, interpolation='nearest', cmap=plt.cm.Blues)

plt.title('Confusion Matrix')

plt.colorbar()

classes = ['Benign', 'Malignant']

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes)

plt.yticks(tick\_marks, classes)

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.show()

# ROC Curve

y\_score = rf\_classifier.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_score)

roc\_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend(loc="lower right")

plt.show()