**TASK 4: Disease Prediction from Medical Data**

**Name:U.Bheemesh**

**Domain:** **Machine Learning**

**Source Code :**

**# Disease Prediction from Medical Data**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**# Machine Learning Models**

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

**from sklearn.preprocessing import StandardScaler**

**from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.svm import SVC**

**from sklearn.ensemble import RandomForestClassifier**

**from xgboost import XGBClassifier**

**# Dataset**

**from sklearn.datasets import load\_breast\_cancer**

**# Load Breast Cancer Dataset**

**data = load\_breast\_cancer()**

**df = pd.DataFrame(data.data, columns=data.feature\_names)**

**df['target'] = data.target**

**# Display basic info**

**print("Dataset shape:", df.shape)**

**print("Target classes:", data.target\_names)**

**print(df.head())**

**# -------------------------------**

**# Step 1: Preprocessing**

**# -------------------------------**

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

**y = df['target']**

**# Standardize features**

**scaler = StandardScaler()**

**X\_scaled = scaler.fit\_transform(X)**

**# Train-test split**

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

**# -------------------------------**

**# Step 2: Model Training & Evaluation**

**# -------------------------------**

**models = {**

**"Logistic Regression": LogisticRegression(),**

**"SVM": SVC(kernel='linear', probability=True),**

**"Random Forest": RandomForestClassifier(n\_estimators=100),**

**"XGBoost": XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss')**

**}**

**for name, model in models.items():**

**print(f"\n🧪 Training {name}...")**

**model.fit(X\_train, y\_train)**

**y\_pred = model.predict(X\_test)**

**acc = accuracy\_score(y\_test, y\_pred)**

**print(f"✅ Accuracy: {acc:.4f}")**

**print("📊 Classification Report:\n", classification\_report(y\_test, y\_pred))**

**print("🧾 Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))**

**# -------------------------------**

**# Step 3: Visualization (Optional)**

**# -------------------------------**

**# Feature importance for Random Forest**

**importances = models["Random Forest"].feature\_importances\_**

**indices = np.argsort(importances)[::-1]**

**plt.figure(figsize=(12, 6))**

**sns.barplot(x=importances[indices], y=data.feature\_names[indices])**

**plt.title('Feature Importance (Random Forest)')**

**plt.xlabel('Importance Score')**

**plt.ylabel('Feature')**

**plt.show()**

**Output:**

**Dataset shape: (569, 31)**

**Target classes: ['malignant' 'benign']**

**mean radius mean texture mean perimeter mean area mean smoothness \**

**0 17.99 10.38 122.80 1001.0 0.11840**

**1 20.57 17.77 132.90 1326.0 0.08474**

**2 19.69 21.25 130.00 1203.0 0.10960**

**3 11.42 20.38 77.58 386.1 0.14250**

**4 20.29 14.34 135.10 1297.0 0.10030**

**mean compactness mean concavity mean concave points mean symmetry \**

**0 0.27760 0.3001 0.14710 0.2419**

**1 0.07864 0.0869 0.07017 0.1812**

**2 0.15990 0.1974 0.12790 0.2069**

**3 0.28390 0.2414 0.10520 0.2597**

**4 0.13280 0.1980 0.10430 0.1809**

**mean fractal dimension ... worst texture worst perimeter worst area \**

**0 0.07871 ... 17.33 184.60 2019.0**

**1 0.05667 ... 23.41 158.80 1956.0**

**2 0.05999 ... 25.53 152.50 1709.0**

**3 0.09744 ... 26.50 98.87 567.7**

**4 0.05883 ... 16.67 152.20 1575.0**

**worst smoothness worst compactness worst concavity worst concave points \**

**0 0.1622 0.6656 0.7119 0.2654**

**1 0.1238 0.1866 0.2416 0.1860**

**2 0.1444 0.4245 0.4504 0.2430**

**3 0.2098 0.8663 0.6869 0.2575**

**4 0.1374 0.2050 0.4000 0.1625**

**worst symmetry worst fractal dimension target**

**0 0.4601 0.11890 0**

**1 0.2750 0.08902 0**

**2 0.3613 0.08758 0**

**3 0.6638 0.17300 0**

**4 0.2364 0.07678 0**

**[5 rows x 31 columns]**

**🧪 Training Logistic Regression...**

**✅ Accuracy: 0.9737**

**📊 Classification Report:**

**precision recall f1-score support**

**0 0.98 0.95 0.96 43**

**1 0.97 0.99 0.98 71**

**accuracy 0.97 114**

**macro avg 0.97 0.97 0.97 114**

**weighted avg 0.97 0.97 0.97 114**

**🧾 Confusion Matrix:**

**[[41 2]**

**[ 1 70]]**

**🧪 Training SVM...**

**✅ Accuracy: 0.9561**

**📊 Classification Report:**

**precision recall f1-score support**

**0 0.93 0.95 0.94 43**

**1 0.97 0.96 0.96 71**

**accuracy 0.96 114**

**macro avg 0.95 0.96 0.95 114**

**weighted avg 0.96 0.96 0.96 114**

**🧾 Confusion Matrix:**

**[[41 2]**

**[ 3 68]]**

**🧪 Training Random Forest...**

**✅ Accuracy: 0.9649**

**📊 Classification Report:**

**precision recall f1-score support**

**0 0.98 0.93 0.95 43**

**1 0.96 0.99 0.97 71**

**accuracy 0.96 114**

**macro avg 0.97 0.96 0.96 114**

**weighted avg 0.97 0.96 0.96 114**

**🧾 Confusion Matrix:**

**[[40 3]**

**[ 1 70]]**

**🧪 Training XGBoost...**

**/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:38:13] WARNING: /workspace/src/learner.cc:740:**

**Parameters: { "use\_label\_encoder" } are not used.**

**warnings.warn(smsg, UserWarning)**

**✅ Accuracy: 0.9561**

**📊 Classification Report:**

**precision recall f1-score support**

**0 0.95 0.93 0.94 43**

**1 0.96 0.97 0.97 71**

**accuracy 0.96 114**

**macro avg 0.96 0.95 0.95 114**

**weighted avg 0.96 0.96 0.96 114**

**🧾 Confusion Matrix:**

**[[40 3]**

**[ 2 69]]**

