

ASSIGNMENT 4 - CLASSIFICATION

Objective:

The objective of this assessment is to evaluate your understanding and ability to apply supervised learning techniques to a real-world dataset.

Dataset:

Use the breast cancer dataset available in the sklearn library.

```
In [3]: import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

1. Loading and Preprocessing

- Load the breast cancer dataset from sklearn.
- Preprocess the data to handle any missing values and perform necessary feature scaling.
- Explain the preprocessing steps you performed and justify why they are necessary for this dataset.

```
In [7]: # Importing necessary Libraries
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np
```

```
In [9]: # Load the breast cancer dataset
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target, name="target")
```

```
# Display basic info about the dataset
print("Dataset shape:", X.shape)
print("Number of missing values in each column:\n", X.isnull().sum())

# Preprocessing: Handling missing values
# In this dataset, there are no missing values. However, in general, we can handle missing values as follows:
# Example: X.fillna(X.mean(), inplace=True) # Replace missing values with column mean

# Preprocessing: Feature scaling
scaler = StandardScaler() # Standardization scales the features to have mean=0 and variance=1
X_scaled = scaler.fit_transform(X) # Fit and transform the features

# Splitting the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Verify scaled data
print("\nFirst 5 rows of scaled features:\n", pd.DataFrame(X_scaled, columns=data.feature_names).head())
```

Dataset shape: (569, 30)

Number of missing values in each column:

mean radius	0
mean texture	0
mean perimeter	0
mean area	0
mean smoothness	0
mean compactness	0
mean concavity	0
mean concave points	0
mean symmetry	0
mean fractal dimension	0
radius error	0
texture error	0
perimeter error	0
area error	0
smoothness error	0
compactness error	0
concavity error	0
concave points error	0
symmetry error	0
fractal dimension error	0
worst radius	0
worst texture	0
worst perimeter	0
worst area	0
worst smoothness	0
worst compactness	0
worst concavity	0
worst concave points	0
worst symmetry	0
worst fractal dimension	0

dtype: int64

First 5 rows of scaled features:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	\
0	1.097064	-2.073335	1.269934	0.984375	1.568466	
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	
2	1.579888	0.456187	1.566503	1.558884	0.942210	
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	
4	1.750297	-1.151816	1.776573	1.826229	0.280372	

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	3.283515	2.652874	2.532475	2.217515	
1	-0.487072	-0.023846	0.548144	0.001392	
2	1.052926	1.363478	2.037231	0.939685	
3	3.402909	1.915897	1.451707	2.867383	
4	0.539340	1.371011	1.428493	-0.009560	

	mean fractal dimension	...	worst radius	worst texture	worst perimeter	\
0	2.255747	...	1.886690	-1.359293	2.303601	
1	-0.868652	...	1.805927	-0.369203	1.535126	
2	-0.398008	...	1.511870	-0.023974	1.347475	
3	4.910919	...	-0.281464	0.133984	-0.249939	
4	-0.562450	...	1.298575	-1.466770	1.338539	

	worst area	worst smoothness	worst compactness	worst concavity	\
0	2.001237	1.307686	2.616665	2.109526	
1	1.890489	-0.375612	-0.430444	-0.146749	
2	1.456285	0.527407	1.082932	0.854974	
3	-0.550021	3.394275	3.893397	1.989588	
4	1.220724	0.220556	-0.313395	0.613179	

	worst concave points	worst symmetry	worst fractal dimension	
0	2.296076	2.750622	1.937015	
1	1.087084	-0.243890	0.281190	
2	1.955000	1.152255	0.201391	
3	2.175786	6.046041	4.935010	
4	0.729259	-0.868353	-0.397100	

[5 rows x 30 columns]

2. Classification Algorithm Implementation

- Implement the following five classification algorithms:

1. Logistic Regression
2. Decision Tree Classifier
3. Random Forest Classifier
4. Support Vector Machine (SVM)
5. k-Nearest Neighbors (k-NN)

- For each algorithm, provide a brief description of how it works and why it might be suitable for this dataset

```
In [17]: # Importing necessary Libraries
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score

# Dictionary to store classifiers and their names
classifiers = {
    "Logistic Regression": LogisticRegression(random_state=42),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42, n_estimators=100),
    "Support Vector Machine (SVM)": SVC(kernel='linear', random_state=42),
    "k-Nearest Neighbors (k-NN)": KNeighborsClassifier(n_neighbors=5)
}

# Loop through each classifier, train it, and evaluate it
for name, clf in classifiers.items():
    # Train the classifier
    clf.fit(X_train, y_train)
    # Make predictions
    y_pred = clf.predict(X_test)
    # Evaluate performance
    print(f"\n== {name} ==")
    print("Accuracy: {:.2f}%".format(accuracy_score(y_test, y_pred) * 100))
    print("Classification Report:\n", classification_report(y_test, y_pred))
```

==== Logistic Regression ===

Accuracy: 97.37%

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

==== Decision Tree ===

Accuracy: 94.74%

Classification Report:

	precision	recall	f1-score	support
0	0.93	0.93	0.93	43
1	0.96	0.96	0.96	71
accuracy			0.95	114
macro avg	0.94	0.94	0.94	114
weighted avg	0.95	0.95	0.95	114

==== Random Forest ===

Accuracy: 96.49%

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

==== Support Vector Machine (SVM) ===

Accuracy: 95.61%

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

==== k-Nearest Neighbors (k-NN) ===

Accuracy: 94.74%

Classification Report:

	precision	recall	f1-score	support
0	0.93	0.93	0.93	43
1	0.96	0.96	0.96	71
accuracy			0.95	114
macro avg	0.94	0.94	0.94	114
weighted avg	0.95	0.95	0.95	114

Explanation of Each Classification Algorithm:

Logistic Regression:

How it works: Logistic regression is a linear model used for binary classification. It calculates the probability of an instance belonging to a class using the sigmoid function and makes predictions based on a decision boundary (typically 0.5).

Suitability: Since the breast cancer dataset is a binary classification problem (malignant or benign), logistic regression is a strong choice due to its simplicity and effectiveness for linearly separable data.

Decision Tree Classifier:

How it works: A decision tree splits the dataset into subsets based on feature values, creating a tree structure. Each node represents a feature, and branches represent decisions leading to leaf nodes (classes).

Suitability: Decision trees handle non-linear relationships well and are interpretable. They are suitable for datasets with complex interactions between features, as seen in the breast cancer dataset.

Random Forest Classifier:

How it works: A random forest is an ensemble method that trains multiple decision trees on random subsets of the data and aggregates their predictions through majority voting.

Suitability: Random forests reduce overfitting (common in single decision trees) and improve generalization, making them effective for datasets with noise and feature interactions like this one.

Support Vector Machine (SVM):

How it works: SVM creates a hyperplane that best separates the data into classes by maximizing the margin between the closest points of each class (support vectors).

Suitability: SVM works well for datasets with a clear margin of separation between classes and performs robustly with high-dimensional data like the breast cancer dataset.

k-Nearest Neighbors (k-NN):

How it works: k-NN classifies an instance based on the majority class among its k nearest neighbors in the feature space.

Suitability: k-NN is non-parametric and works well for datasets where the decision boundary is not linear. It can capture complex patterns in the breast cancer dataset.

3. Model Comparison

- Compare the performance of the five classification algorithms.
- Which algorithm performed the best and which one performed the worst?

Comparison of Classification Algorithms

1. Logistic Regression

Accuracy: 97.37%

Precision: High for both classes (0 = 98%, 1 = 97%)

Recall: High for both classes (0 = 95%, 1 = 99%)

F1-Score: 96% (Class 0), 98% (Class 1)

Observations: Logistic regression performed the best, with the highest accuracy (97.37%) and strong precision, recall, and F1-scores. It is particularly well-suited for this dataset because it works effectively for binary classification problems with linearly separable data.

2. Decision Tree

Accuracy: 94.74%

Precision: Good for both classes (0 = 93%, 1 = 96%)

Recall: Good for both classes (0 = 93%, 1 = 96%)

F1-Score: 93% (Class 0), 96% (Class 1)

Observations: Decision trees performed decently but had the lowest accuracy (94.74%) among the models. While they are good at capturing non-linear relationships, they are prone to overfitting, which may explain the slightly lower performance compared to other models.

3. Random Forest

Accuracy: 96.49%

Precision: High for both classes (0 = 98%, 1 = 96%)

Recall: High for both classes (0 = 93%, 1 = 99%)

F1-Score: 95% (Class 0), 97% (Class 1)

Observations: Random forests performed almost as well as logistic regression. They handled feature interactions and reduced overfitting, which helped them achieve high precision and recall. However, their slightly lower recall for Class 0 compared to logistic

regression might explain the slightly lower accuracy.

4. Support Vector Machine (SVM)

Accuracy: 95.61%

Precision: Good for both classes (0 = 93%, 1 = 97%)

Recall: Good for both classes (0 = 95%, 1 = 96%)

F1-Score: 94% (Class 0), 96% (Class 1)

Observations: SVM also performed well, with an accuracy of 95.61%. Its ability to find a decision boundary with a large margin likely contributed to its good performance. However, its accuracy was slightly lower than logistic regression and random forests.

5. k-Nearest Neighbors (k-NN)

Accuracy: 94.74%

Precision: Good for both classes (0 = 93%, 1 = 96%)

Recall: Good for both classes (0 = 93%, 1 = 96%)

F1-Score: 93% (Class 0), 96% (Class 1)

Observations: k-NN performed similarly to the decision tree, with an accuracy of 94.74%. While it is simple and effective, it may be sensitive to the choice of k and feature scaling. The slightly lower accuracy and F1-scores indicate it might not capture complex patterns as effectively as the other algorithms.

Best and Worst-Performing Algorithms

Best-Performing Algorithm: Logistic Regression

Logistic regression achieved the highest accuracy (97.37%) and strong performance across precision, recall, and F1-scores for both classes. This is likely because the dataset is well-suited for linear classification.

Worst-Performing Algorithm: Decision Tree (and k-NN)

Decision tree and k-NN both had the lowest accuracy (94.74%). Decision trees may have suffered from overfitting, while k-NN might have struggled with the optimal k and distance-based classification.