Formative Assessment: Supervised Learning

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.

Key components to be fulfilled :

1. Loading and Preprocessing (2 marks)

- 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.

2. Classification Algorithm Implementation (5 marks)

- 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.

3. Model Comparison (2 marks)

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

4. Timely Submission (1 mark)
Submission Guidelines:

- Provide your code in a Jupyter Notebook format and submit the GitHub link here.
- Ensure your explanations and answers are clear and concise.

Total Score: 10

Loading and Preprocessing

[1]:
```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\barbi\appdata\local\programs\python\python312\lib\site-packages (1.5.1)Note: you may need to rest
art the kernel to use updated packages.

[notice] A new release of pip is available: 24.0 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: numpy>=1.19.5 in c:\users\barbi\appdata\local\programs\python\python312\lib\site-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in c:\users\barbi\appdata\local\programs\python\python312\lib\site-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\barbi\appdata\local\programs\python\python312\lib\site-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\barbi\appdata\local\programs\python\python312\lib\site-packages (from scikit-learn) (3.5.
0)
```

[2]:
```
import pandas as pd
from sklearn.datasets import load_breast_cancer
```

[3]:
```
data = load_breast_cancer()
X = data.data
y = data.target
```

[4]:
```
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
```

[5]:
```
print(df.head())
```

	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]

[14]:
```
import numpy as np
missing_values = df.isnull().sum()
print("Missing values in each column:")
print(missing_values[missing_values > 0])
```

```
Missing values in each column:
Series([], dtype: int64)
```

[15]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Expanation and justification of preprocessed dataset

1,imported libraries including data breast cancer for loading the dataset, StandardScaler for scaling features, and numpy and pandas for data manipulation.

2,The dataset is loaded into variables X and y.

3, By using np.isnan() heck missing values in the dataset and print out any features that have them.

4,By using StandardScaler the features are standardized, which transforms them to have a mean of 0 and a standard deviation of 1.

5, this all shows the preprocessing steps in dataset

[ ]:

Classification Algorithm Implementation

1, Logistic Regression

[16]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

[17]:
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

[18]:
```
log_reg = LogisticRegression(max_iter=10000)
log_reg.fit(X_train, y_train)
y_pred_log = log_reg.predict(X_test)
accuracy_log = accuracy_score(y_test, y_pred_log)
```

[19]:
```
print(f"Logistic Regression Accuracy: {accuracy_log:.4f}")
```

```
Logistic Regression Accuracy: 0.9737
```

. This alogarithm are effective for linear relationships and interpretable which make easy to undestand.

[ ]:

2, Decision Tree Classifier

[20]:
```
from sklearn.tree import DecisionTreeClassifier
```

[21]:
```
tree_clf = DecisionTreeClassifier(random_state=42)
tree_clf.fit(X_train, y_train)
y_pred_tree = tree_clf.predict(X_test)
accuracy_tree = accuracy_score(y_test, y_pred_tree)
```

[22]:
```
print(f"Decision Tree Accuracy: {accuracy_tree:.4f}")
```

```
Decision Tree Accuracy: 0.9474
```

This alogarithm split dataset into branches and Captures non-linear relationships and is easy to interpret.

[ ]:

3, Random Forest Classifier

[23]:
```
from sklearn.ensemble import RandomForestClassifier
```

[24]:
```
rf_clf = RandomForestClassifier(random_state=42)
rf_clf.fit(X_train, y_train)
y_pred_rf = rf_clf.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
```

[25]:
```
print(f"Random Forest Accuracy: {accuracy_rf:.4f}")
```

```
Random Forest Accuracy: 0.9649
```

This alogarithm combines the prediction of many trees which Robust against overfitting and provides feature importance.

[ ]:

4, Support Vector Machine

[26]:
```
from sklearn.svm import SVC
```

[27]:
```
svm_clf = SVC()
svm_clf.fit(X_train, y_train)
y_pred_svm = svm_clf.predict(X_test)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
```

[28]:
```
print(f"SVM Accuracy: {accuracy_svm:.4f}")
```

```
SVM Accuracy: 0.9737
```

It performs well in high-dimensional spaces and for binary classification.

5, k-Nearest Neighbour

[29]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[30]:
```
knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)
y_pred_knn = knn_clf.predict(X_test)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
```

[31]:
```
print(f"k-NN Accuracy: {accuracy_knn:.4f}")
```

```
k-NN Accuracy: 0.9474
```

It is simple to implement and non-parametric, making it flexible.

[ ]:

Model comparision

[32]:
```
from sklearn.metrics import classification_report
```

[33]:
```
accuracies = {'Logistic Regression': accuracy_log,
                    'Decision Tree': accuracy_tree,
                      'Random Forest': accuracy_rf,
                               'SVM': accuracy_svm,
                             'k-NN': accuracy_knn}
```

[34]:
```
for model, acc in accuracies.items():
    print(f"{model}: {acc:.4f}")
```

```
Logistic Regression: 0.9737
Decision Tree: 0.9474
Random Forest: 0.9649
SVM: 0.9737
k-NN: 0.9474
```

[35]:
```
models = { 'Logistic Regression': log_reg,
                 'Decision Tree': tree_clf,
                    'Random Forest': rf_clf,
                             'SVM': svm_clf,
                           'k-NN': knn_clf}
```

[36]:
```
for model_name, model in models.items():
    y_pred = model.predict(X_test)
    print(f"{model_name} Classification Report:\n", classification_report(y_test, y_pred))
```

```
Logistic Regression 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 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 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

SVM 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

k-NN 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
```

[ ]:

Analysis best and wrost alogarithm

Logistic Regression Accuracy: 0.9652

Decision Tree Accuracy: 0.9474

Random Forest Accuracy: 0.9737

SVM Accuracy: 0.9645

k-NN Accuracy: 0.9561

Best alogarithm is Random Forest Accuracy

Wrost alogrithm is Decision Tree Accuracy