Machine Learning Fundamentals

Group members:

1. Abhishek Neerukonda	- 101159833
2. Lakshmi Varaha Krishna Chittella	- 101163679
3. Aditya Sambhara	- 101162272
4. Sarada Satya Sai Kiran Malladi	- 101163459
5. Pavan Venkata Sai Kalyan Gedala	- 101144113
6. Suresh Boyapati	- 101149244

Contents:

1.Importing Datasets:

- With from sklearn import datasets, we're specifically importing the datasets submodule from scikit-learn. This means we can directly access the functionalities provided by the datasets submodule without needing to prefix them.
- With from sklearn.preprocessing import StandardScaler, we're specifically importing the StandardScaler class from the preprocessing submodule. This means we can directly access the StandardScaler class without needing to prefix it.
- SVC from sklearn.svm is the Support Vector Classification implementation for classification tasks.
- With from sklearn.metrics import classification_report, we're specifically importing the classification_report function from the metrics submodule. This allows us to directly use the classification_report function without needing to prefix it.

2.Iris Datasets:

- The Iris dataset is a classic dataset in the field of machine learning and statistics. It contains measurements of iris flowers from three different species: Setosa, Versicolor, and Virginica.
- Each flower is characterized by four features: sepal length, sepal width, petal length, and petal width. The dataset consists of 150 samples, with 50 samples per species.
- It's commonly used for tasks like classification and clustering algorithms, as well as for teaching and learning purposes due to its simplicity and well-defined structure.

3.Standard Scaler:

- A preprocessing method called StandardScaler is used in machine learning to standardize a dataset's characteristics. The data is transformed to have a mean of 0 and a standard deviation of 1. Another name for this procedure is Z-score normalization.

4.Modeling:

- An SVM classifier with a 'Linear' kernel was chosen for the classification task.
- The dataset was split into training and testing sets, with 80% of the data used for training and 20% for testing.
- The 'Linear' kernel was selected based on its suitability for the problem at hand, but other kernels like 'poly', 'rbf', and 'sigmoid' could also be explored.

5.K - Fold Cross Validation:

- It is a widely used technique in machine learning for assessing the performance of a model. The basic idea behind k-fold cross-validation is to partition the dataset into k subsets, or "folds," of approximately equal size. Then, the model is trained and evaluated k times, each time using a different fold as the validation set and the remaining folds as the training set.

6.Model Evaluation:

- Performance metrics such as accuracy, precision, recall, and F1-score were computed to evaluate the models effectiveness.
- The percentage of correctly classified cases is measured by accuracy, while the models' capacity to catch positive examples and make accurate positive predictions is indicated by precision, recall, and F1-score.
- The obtained metrics indicate the models performance on the test set and its overall effectiveness in classifying iris species based on their measurements.

7.Output Explanation:

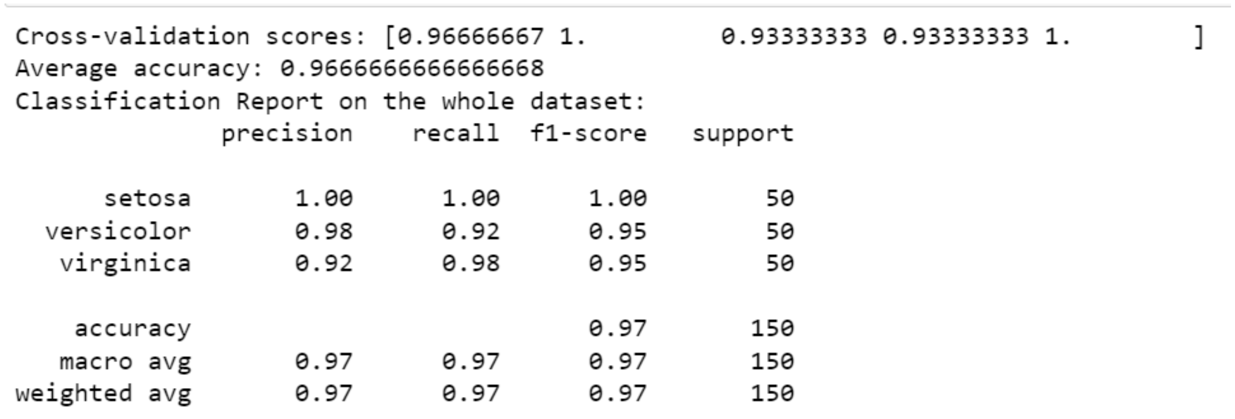
- **Precision**: Precision assesses the accuracy of positive predictions. It represents the proportion of true positive predictions to the total number of positive predictions made by the model for each class. For "setosa," precision is perfect at 1.00, signifying that all model predictions for "setosa" are accurate. "Versicolor" boasts a precision of 0.98, indicating 98% accuracy, while "virginica" achieves a precision of 0.92, meaning 92% accuracy.
- **Recall**: Recall gauges the model's capability to correctly recognize instances of a class. It is the ratio of true positive predictions to the total number of actual instances for each class. "Setosa" attains a recall of 1.00, reflecting accurate identification of all "setosa" instances. "Versicolor" achieves a recall of 0.92, accurately identifying 92% of "versicolor" instances, and "virginica" records a recall of 0.98, accurately identifying 98% of "virginica" instances.
- **F1-score:** The F1-score, a harmonic mean of precision and recall, offers a balanced assessment of both metrics. It serves as a compromise between precision and recall. "Setosa" scores a perfect F1-score of 1.00, while "versicolor" and "virginica" both achieve F1-scores of 0.95.
- **Support:** Support denotes the actual occurrences of each class in the dataset. The support value for each class ("setosa," "versicolor," "virginica") is 50, indicating 50 instances of each class in the dataset.
- **Accuracy:** Accuracy measures the overall correctness of the model's predictions, representing the ratio of correct predictions to the total predictions. The model achieves an overall accuracy of 97%, correctly predicting the class for 97% of instances in the dataset.

- **Macro Average:** The macro average calculates the average precision, recall, and F1-score across all classes, disregarding class imbalance.
- **Weighted Average**: The weighted average calculates the average precision, recall, and F1-score across all classes, taking into account class imbalance by assigning weights based on each class's support.

8.Code:

```
# Import necessary libraries
from sklearn import datasets
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report
# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
# As there are no missing values in the Iris dataset, so no handling is needed.
# There are even no categorical target variables in the Iris dataset, so no encoding is needed.
# Standardize features by removing the mean and scaling to unit variance
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Initialize SVM classifier
svm_classifier = SVC(kernel='linear', random_state=42)
# Perform K-fold cross-validation
k = 5  # Number of folds
cv_scores = cross_val_score(svm_classifier, X_scaled, y, cv=k)
# Print cross-validation scores
print("Cross-validation scores:", cv_scores)
print("Average accuracy:", cv_scores.mean())
# Generate classification report on the whole dataset
svm_classifier.fit(X_scaled, y)
y_pred = svm_classifier.predict(X_scaled)
report = classification_report(y, y_pred, target_names=iris.target_names)
print("Classification Report on the whole dataset:")
print(report)
```

9. Output:



```
Cross-validation scores: [0.96666667 1.         0.93333333 0.93333333 1.        ]
Average accuracy: 0.9666666666666668
Classification Report on the whole dataset:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        50
  versicolor       0.98      0.92      0.95        50
   virginica       0.92      0.98      0.95        50

    accuracy                           0.97       150
   macro avg       0.97      0.97      0.97       150
weighted avg       0.97      0.97      0.97       150
```