Task 10

Artificial Intelligence with Python

Semester 6th, Spring 2024

**Task Title: Implementing Complete ML Project Using Iris dataset**

In this project, we are classifying flowers into three species (setosa, versicolor, and virginica) based on their features (sepal length, sepal width, petal length, and petal width).

**Solution:**

**Explanation of the code**

You would have to install **pandas** and **seaborn** to run the project successfully. For installing them in thorn, please click tools - manage packages in thorn, search pandas and install. Perform same steps for seaborn.

### Step 1: Data Collection

Explanation:

1. Import necessary libraries: We import the load\_iris function from sklearn.datasets and pandas for data manipulation.
2. Load the Iris dataset: load\_iris() loads the dataset into a dictionary-like object.
3. Create a DataFrame: We convert the data into a pandas DataFrame for easier handling and label the columns with feature names.
4. Add target variable: Add the target variable (species) to the DataFrame.
5. Display data: Print the first few rows to inspect the dataset.

### Step 2: Data Preprocessing

Explanation:

1. Import necessary libraries: We import train\_test\_split for splitting the dataset and StandardScaler for feature scaling.
2. Separate features and target: X contains the features, and y contains the target variable (species).
3. Split data: train\_test\_split divides the data into training and testing sets (80% train, 20% test) with a fixed random state for reproducibility.
4. Standardize features: StandardScaler standardizes the features by removing the mean and scaling to unit variance. fit\_transform is applied to the training set and transform to the test set.

### Step 3: Exploratory Data Analysis (EDA)

Explanation:

1. Import necessary libraries: We import matplotlib.pyplot and seaborn for visualization.
2. Pairplot: sns.pairplot creates pairwise scatter plots to visualize relationships between features, colored by species.
3. Show plot: plt.show() displays the plot.
4. Feature distribution: data.hist creates histograms for each feature to inspect their distributions.

### Step 4: Feature Selection

1. Import necessary library: Import RandomForestClassifier from sklearn.ensemble.
2. Fit model: Instantiate and fit a Random Forest model to the training data.
3. Feature importances: Retrieve the importance of each feature using model.feature\_importances\_.
4. Create DataFrame: Create a DataFrame to display the feature importances.
5. Sort features: Sort the features by importance in descending order.
6. Display feature importances: Print the sorted DataFrame.

### Step 5: Model Selection

Explanation:

1. Import necessary libraries: Import LogisticRegression, RandomForestClassifier, and SVC from sklearn.
2. Initialize models: Create a dictionary of models to compare, with names as keys and model instances as values. Set a higher max\_iter for LogisticRegression to ensure convergence.

### Step 6: Model Training

Explanation:

1. Train models: Loop through the dictionary and train each model using fit on the training data (X\_train, y\_train).

### Step 7: Model Evaluation

Explanation:

1. Import necessary metrics: Import accuracy\_score, precision\_score, recall\_score, f1\_score, and classification\_report from sklearn.metrics.
2. Evaluate models: Loop through the dictionary, make predictions on the test data, and compute evaluation metrics for each model.
3. Print results: Display accuracy, precision, recall, F1 score, and the detailed classification report for each model.

### Step 8: Hyperparameter Tuning

Explanation:

1. Import necessary library: Import GridSearchCV from sklearn.model\_selection.
2. Define parameter grid: Create a dictionary specifying the hyperparameters and their possible values.
3. Initialize GridSearchCV: Instantiate GridSearchCV with RandomForestClassifier, parameter grid, 5-fold cross-validation, and accuracy as the scoring metric.
4. Fit GridSearchCV: Fit the grid search to the training data.
5. Best parameters and score: Print the best hyperparameters and the corresponding score.
6. Train with best parameters: Retrieve the best model and fit it to the training data.

### Summary

This project demonstrates the steps involved in a machine learning pipeline using the Iris dataset, including data collection, preprocessing, exploratory data analysis, feature selection, model selection, training, evaluation, and hyperparameter tuning. Each step is crucial for building a robust and reliable machine learning model. This structured approach helps ensure that the model generalizes well to unseen data and provides accurate predictions.