Question2 :

Using the Forest cover-types dataset from Scikit-learn, implement a classification model based on the following directions -

- Implement basic dataset preprocessing (if required), like imputing

NULL values, encoding categorical variables, scaling numerical

variables, etc.

- EDA (with proper inference from the dataset)
- Train-Test-Split - Use the following as the parameters of

train_test_split to take a train set and test set from the

above-mentioned 3 different vectorized datasets.

    - test_size - 0.2
    - stratify - True
    - random_state - 21
    - shuffle - True
    - train_size - 0.8

- Implement a simple ML model on the train set
- Implement a DL model on the train set
- Evaluation Metrics - Using the test set of the above-mentioned 3

different vectorized datasets test all the above-mentioned models

on the following metrics.

    - Accuracy
    - Precision

- ◯ Recall
- ◯ F1 Score

Steps:

- ## Python version used
  - 3.10.0
- ## Platform used for implementation:
  - Google colab for EDA & finding the Best performing Model
  - Pycharm for doing the end to end pipeline implementation.
    - Building webapp using the Flask & HTML & css
- ## Intial thought Process
  - Step-by-step process to implement the classification model for the given problem statement:
  - ### 1. Data EDA:
    - \- Load the Forest cover-types dataset from Scikit-learn/ download the data from the given link.
    - \- Perform exploratory data analysis to understand the dataset's structure, features, and target variable.
    - \- Analyze the distribution of the target variable (cover types) to check for class imbalances.

- - Explore the statistical summary of numerical features and the frequency distribution of categorical features.
- - Visualize the data using plots, histograms, box plots, etc., to gain insights.

○ 2. Missing Value Imputation / Handling Missing Values:

- - Check for missing values in the dataset.
- - Decide on a suitable strategy for handling missing values (e.g., imputing with mean, median, mode, or dropping rows/columns).
- - Implement the chosen strategy to handle missing values in the dataset.

○ 3. Dataset Preprocessing:

- - Encode categorical variables using techniques like one-hot encoding or label encoding.
- - Scale numerical variables using techniques like Min-Max scaling or Standardization.
- - Split the dataset into features (X) and target (y) variables.

○ 4. Train-Test-Split:

- - Use the `train_test_split` function to split the dataset into a training set and a test set.

- - Set the `test_size` parameter to 0.2, `stratify` to True (to maintain class distribution in both sets), `random_state` to 21 for reproducibility, and `shuffle` to True.
- - Ensure that 80% of the data is used for training (`train_size` set to 0.8).

○ 5. Implement a Simple ML Model:

- - Choose a simple machine learning model (e.g., Logistic Regression, Decision Tree, Random Forest, etc.).
- - Train the chosen model on the training dataset.
- - Make predictions on the test dataset.
- - Evaluate the model's performance using accuracy, precision, recall, and F1 score.

○ 6. Implement a DL Model:

- - Choose a deep learning model (e.g., Neural Network, CNN, LSTM, etc.).
- - Design the architecture of the deep learning model.
- - Compile the model with appropriate loss function and optimizer.
- - Train the deep learning model on the training dataset.
- - Make predictions on the test dataset.

- - Evaluate the model's performance using accuracy, precision, recall, and F1 score.

○ 7. Evaluation Metrics:

- - Evaluate all the models (simple ML and DL models) on the test dataset using the following metrics:
- - Accuracy: The proportion of correctly classified instances to the total instances in the test set.
- - Precision: The proportion of true positive predictions to the total positive predictions (measures model's ability not to label as positive a sample that is negative).
- - Recall: The proportion of true positive predictions to the total actual positive instances (measures model's ability to find all the positive samples).
- - F1 Score: The harmonic mean of precision and recall, provides a balanced measure between precision and recall.

○ 8. Selecting the Best Model:

- - Compare the evaluation metrics for the simple ML and DL models.
- - Choose the model with the best overall performance on the test set based on the evaluation metrics.

- 9. Building the End-to-End Pipeline:
  - - Create a data preprocessing pipeline that includes missing value imputation, categorical variable encoding, and numerical variable scaling.
  - - Integrate the selected best model into the pipeline.
  - - Ensure that the pipeline is designed to take raw data as input and produce predictions as output.

- 10. Building the Web App using Flask:
  - - Create a web application using Flask, a Python web framework.
  - - Design the user interface to accept input from users (features) for which predictions need to be made.
  - - Connect the web interface to the previously built end-to-end pipeline.
  - - Allow users to submit their input data, run predictions using the model, and display the results.

- 11. Deploying the Model using Azure CI/CD Pipeline:
  - - Set up a CI/CD (Continuous Integration / Continuous Deployment) pipeline on Azure or any other cloud platform.
  - - Connect the code repository (e.g., GitHub) to the CI/CD pipeline for automatic builds.

- - Define the deployment process, which includes building the web application and deploying it on the chosen platform.
- - Ensure that the pipeline includes testing and quality assurance checks to ensure the model's correctness and reliability.