1. **It has 7 different classes of skin cancer which are listed below:**

    a. Melanocytic nevi
    b. Melanoma
    c. Benign keratosis-like lesions
    d. Basal cell carcinoma
    e. Actinic keratoses
    f. Vascular lesions
    g. Dermatofibroma

2. **Reading & Preprocessing dataset:**

    a. Read the dataset into a suitable data structure, such as a Pandas Data Frame.
    b. Creating dictionary of different cell types.
    c. Mapping of image id to new column path.
    d. Mapping of dx to new column cell type.The 'dx' column contains labels indicating the type of skin lesion for each image.
    e. Changing categorical cell type to numerical value in new cell type_idx(label encoding)

3. **Data Cleaning:**

    a. Checking missing data.
    b. Only age column has missing values so we are replacing data with mean values.
    c. Printing the data type of all columns.

4. **EDA:**

    a. Plot to see distribution of 7 different classes of cell type

    Observation:

    Its seems from the above plot that in this dataset cell type Melanecytic nevi has very large number of instances in comparison to other cell types

    b. Plotting the distribution of localization field.

    Observation:

    It seems back , lower extremity, trunk and upper extremity are heavily compromised regions of skin cancer

c. Now, check the distribution of Age

Observation:

It seems that there are larger instances of patients having age from 30 to 60

d. Visualize age wise distribution of skin cancer types.

Observation:

It seems that skin cancer types 0,1, 3 and 5 which are Melanocytic nevi,dermatofibroma,Basal cell carcinoma and Vascular lesions are not much prevalant below the age of 20 years.

5. **Scaling/Resize the Images:**

   a. The lambda function takes each file path, opens the corresponding image using the Image.open() function, and then resizes the image to a width of 100 pixels and a height of 75 pixels using the resize() function. Finally, the np.asarray() function converts the resized image into a NumPy array. Skin_df['image'] assigns the resulting NumPy array to a new column in the DataFrame called 'image'.
   b. After resizing the images loading of sample images is done.
   c. Took 5 sample images of each cell type
   d. Created 7 rows corresponding to different skin cancer.
   e. The figsize determines the size of the figure in inches.
   f. Showing the images in 300 dpi resolution as category_samples.png.

6. **Preparing Training and Testing data:**

   a. Dividing the data into 80% training and 20% testing

7. **Normalizing the Data:**

   a. Normalizing the x_train, x_test by subtracting from the mean values and then dividing by their standard deviation.

8. **Label Encoding:**

   a. Changing categorical values of cell type to numerical value (one hot encoding)

9. **Splitting Training and Validation Data:**

   a. Further dividing the actual training data into 90% training and 10% validation

10. Applying Deep Learning Model:

A. This code defines a CNN architecture for skin cancer classification, consisting of the input images have a height of 75 pixels, a width of 100 pixels, and three colour channels (RGB) ,alternating convolutional and pooling layers to extract features, followed by fully connected layers for classification. Dropout layers are used to mitigate overfitting, and the softmax activation function provides class probabilities as the output.

   i. **model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', padding='Same')):** This line adds another convolutional layer with similar settings as the previous layer. The input shape is inferred automatically from the previous layer's output.

   ii. **model.add(MaxPool2D(pool_size=(2, 2))):** This line adds a max-pooling layer to the model. It reduces the spatial dimensions of the input by taking the maximum value within each 2x2 region.

   iii. **model.add(Dense(128, activation='relu')):** This line adds a fully connected layer with 128 neurons and a ReLU activation function.

   iv. **model.add(Dropout(0.25)):** This line adds a dropout layer, which randomly sets a fraction of input units to 0 during training to reduce overfitting. In this case, 25% of the units will be dropped.

   v. **model.add(Dense(num_classes, activation='softmax')):** This line adds the final fully connected layer with a number of neurons equal to the number of classes in the dataset. The softmax activation function is used to obtain the probability distribution over the classes.

B. The line of code initializes the Adam optimizer with specific parameter values for learning rate, decay rates, and other optional settings. The optimizer will be used to update the model's parameters during the training process, aiming to optimize the model's performance on the given task.

   i. **lr (learning rate):** It determines the step size at each iteration while updating the parameters of the model. In this case, the learning rate is set to 0.001, which means that the optimizer will make relatively small updates to the parameters.

   ii. **beta_1:** It is the exponential decay rate for the first-moment estimates (mean) of the gradients. The value of 0.9 indicates that the optimizer will consider the previous gradients with a weight of 0.9 when computing the current gradient update.

   iii. **beta_2:** It is the exponential decay rate for the second-moment estimates (variance) of the gradients. The value of 0.999 indicates that the optimizer will consider the previous squared gradients with a weight of 0.999 when computing the current gradient update.

iv. **epsilon:** It is a small constant value added to the denominator for numerical stability. If set to None, the default value will be used.

v. **decay:** It specifies the learning rate decay over each update. A decay value of 0.0 means that the learning rate will remain constant throughout the training process.

vi. **amsgrad:** It is a boolean value that determines whether to use the AMSGrad variant of the Adam algorithm. When set to False (as in this case), the standard Adam algorithm is used.

C. Data Augmentation:

i. Approaches that alter the training data in ways that change the array representation while keeping the label the same are known as data augmentation techniques. By applying just, a couple of these transformations to our training data, we can easily double or triple the number of training examples and create a very robust model.

11. Final Prediction of Cell type: In this step we will check the testing accuracy and validation accuracy of our model, plot confusion matrix and also check the misclassified images count of each type.

a. Prediction on validation data:

i. **Validation: accuracy** = 0.739402 ; **loss_v** = 0.707150

ii. **Test: accuracy** = 0.748377 ; **loss** = 0.706361

iii. **According to confusion matrix**: our model has maximum number of incorrect predictions for Basal cell carcinoma which has code 3, then second most misclassified type is Vascular lesions code 5 then Melanocytic nevi code 0 where as Actinic keratoses code 4 has least misclassified type.

I achieved **74.83%** accuracy on final testing data. We can also further tune our model to achieve the accuracy around or above 80%.