SKIN CANCER IMAGE CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS

Problem Statement

- Considering the limited availability of the resources, early detection of skin cancer is highly important.
- Accurate diagnosis and feasibility of detection are vital in general for skin cancer prevention policy.
- Skin cancer detection in early phases is a challenge for even the dermatologist.

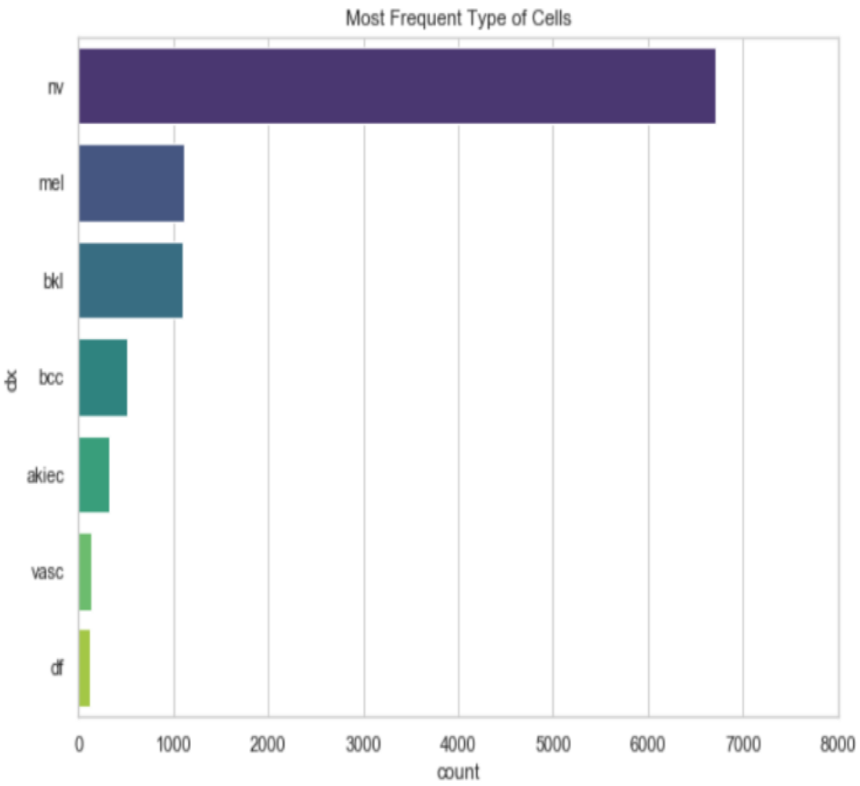
Objective

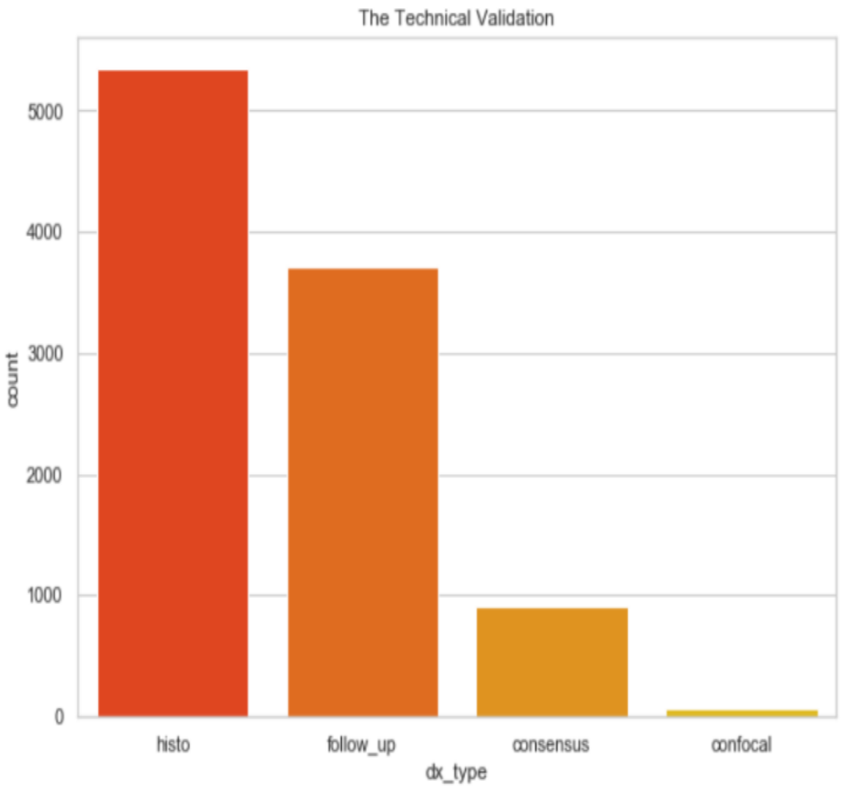
- To detect 7 different classes of skin cancer using Convolution Neural Network with keras tensorflow in backend.
- To analyse the result to see how the model can be useful in practical scenario.
- Diagnosing methodology using Image Processing and Deep Learning models.

Data Description

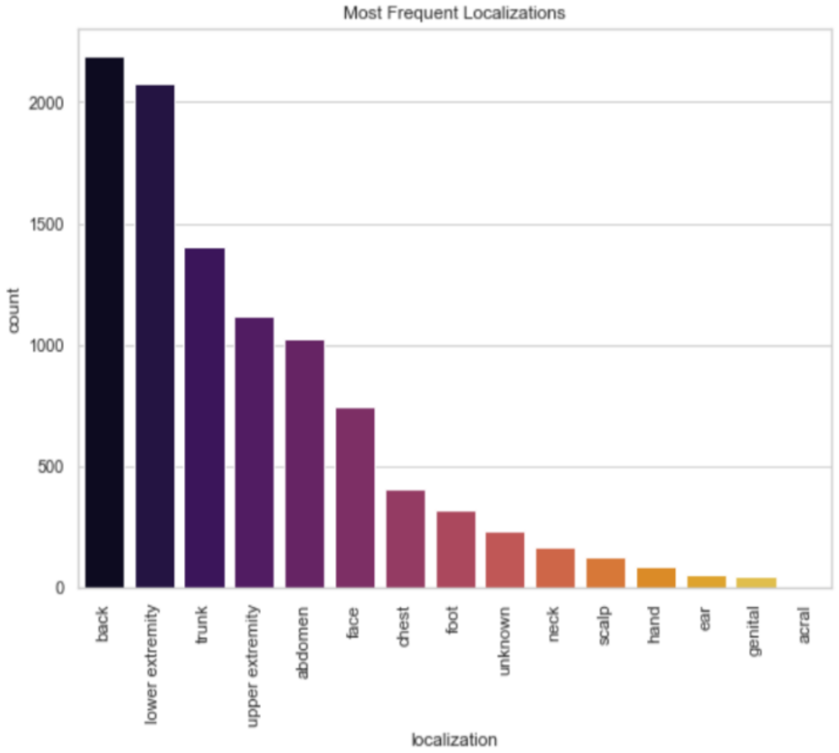
- The dataset that we are using is the HAM10000 dataset.
- 10015 dermatoscopic images.
- 7 features - lesion id, image id, dx, dx_type, age, sex, localization
- It has 7 different classes of skin cancer:
    1. Melanocytic nevi
    2. Melanoma
    3. Benign keratosis
    4. Basal cell carcinoma
    5. Actinic keratoses
    6. Vascular lesions
    7. Dermatofibroma

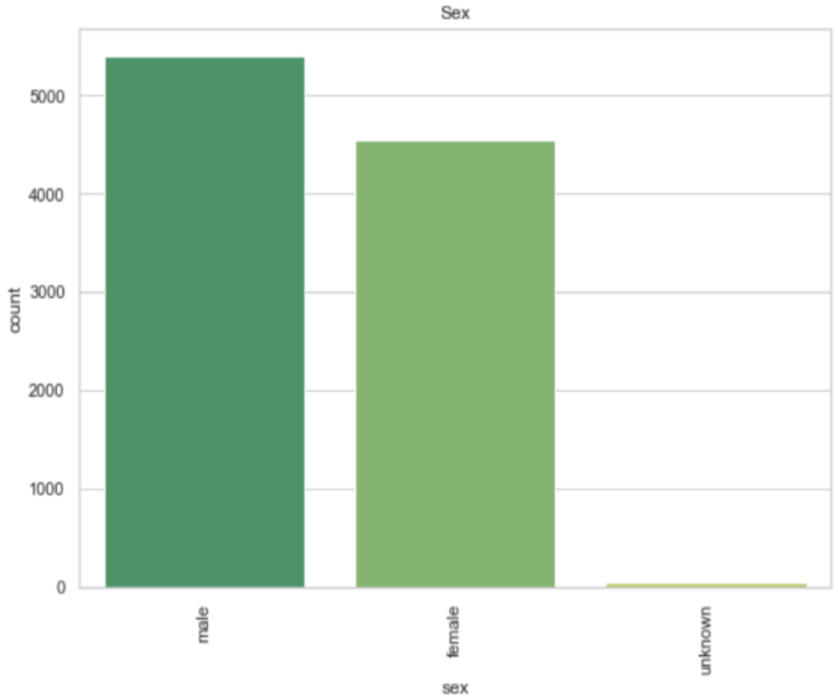
Exploratory Data Analysis - Type of Skin Cancer and Technical Validation



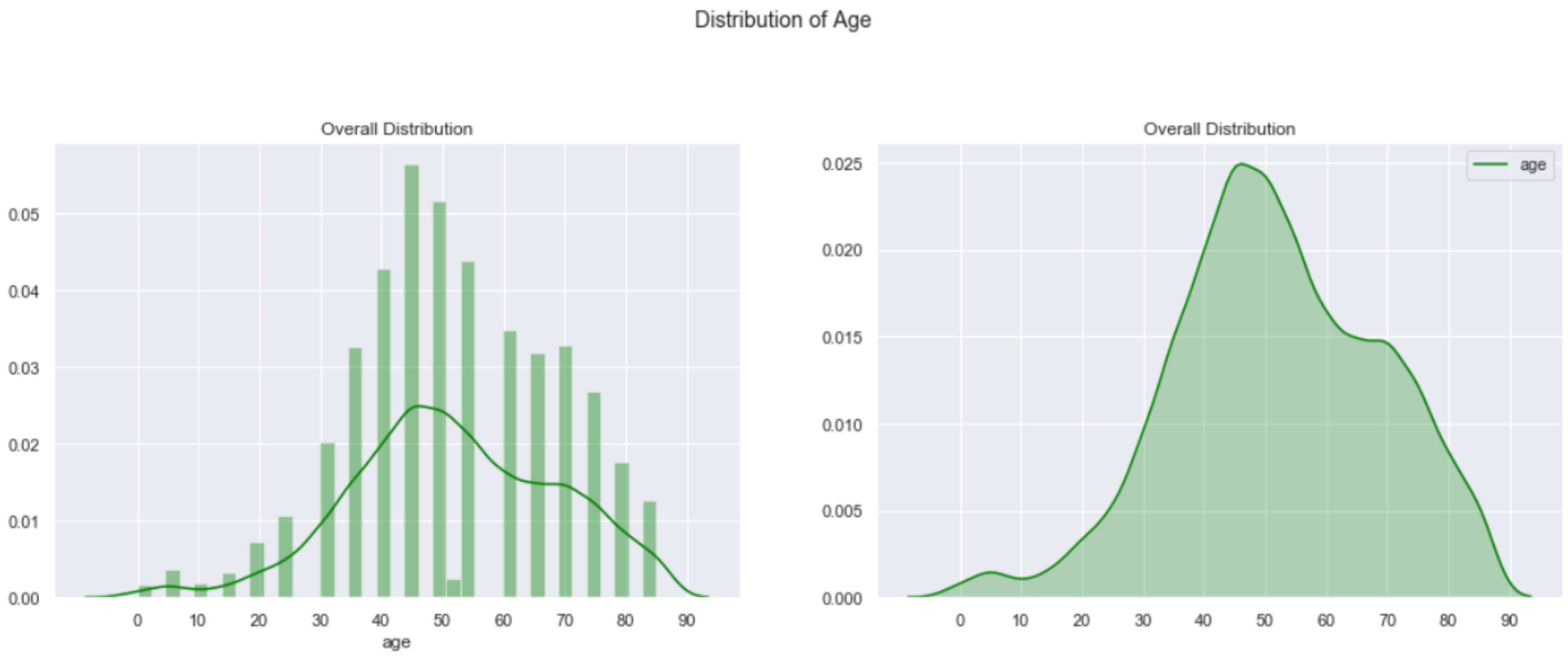


Exploratory Data Analysis - Localization and Sex

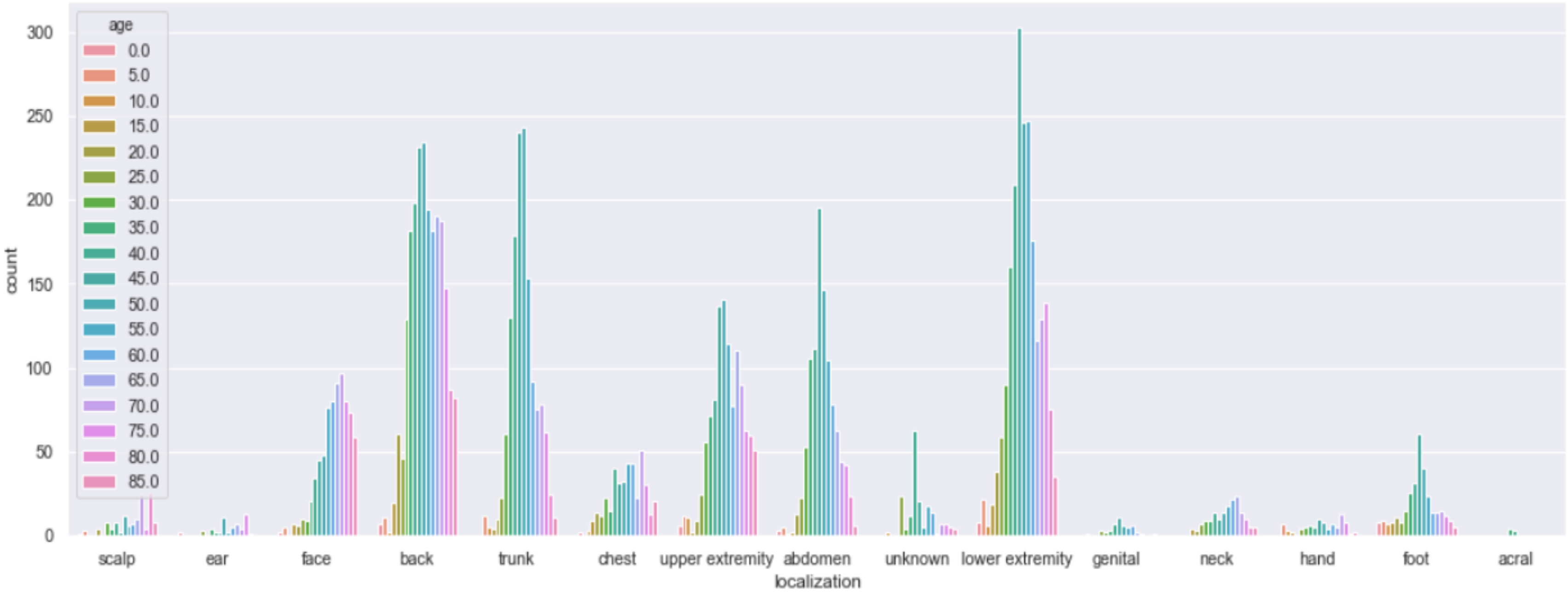




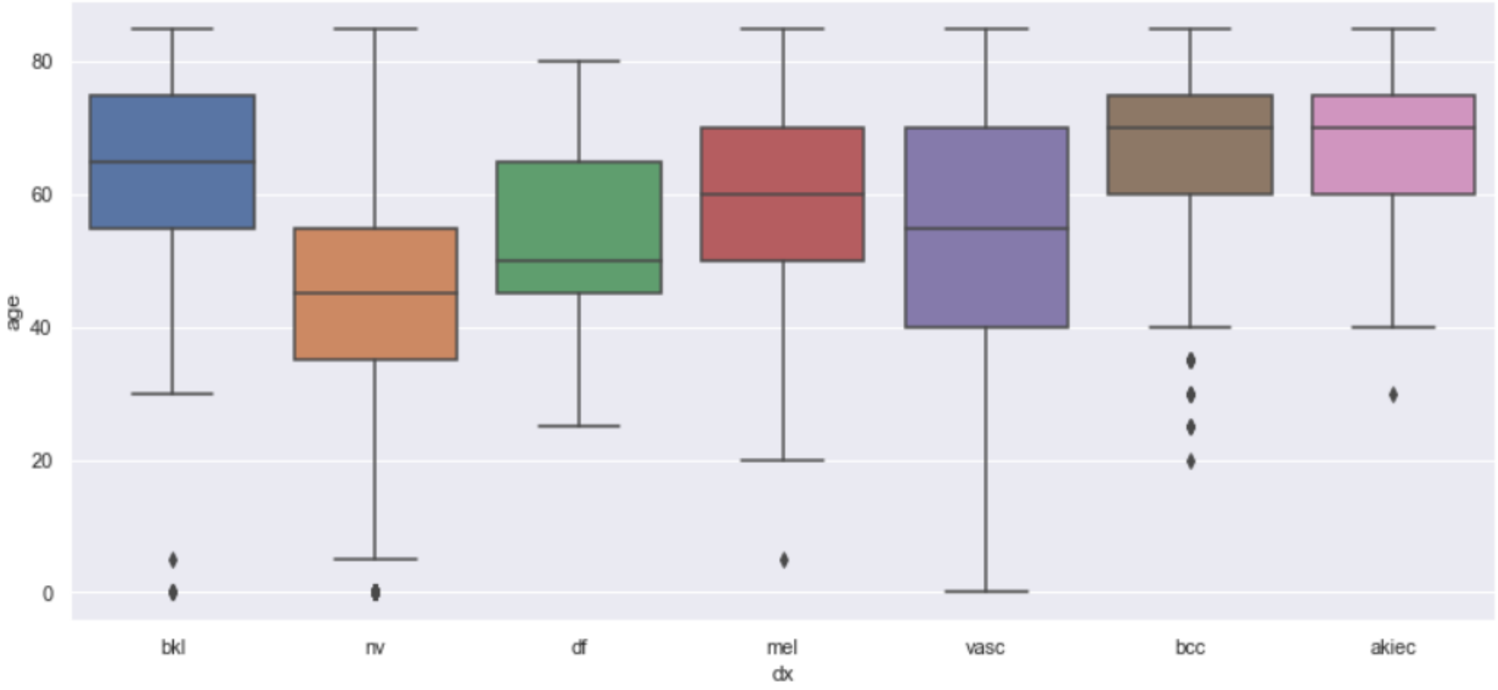
Exploratory Data Analysis - Age



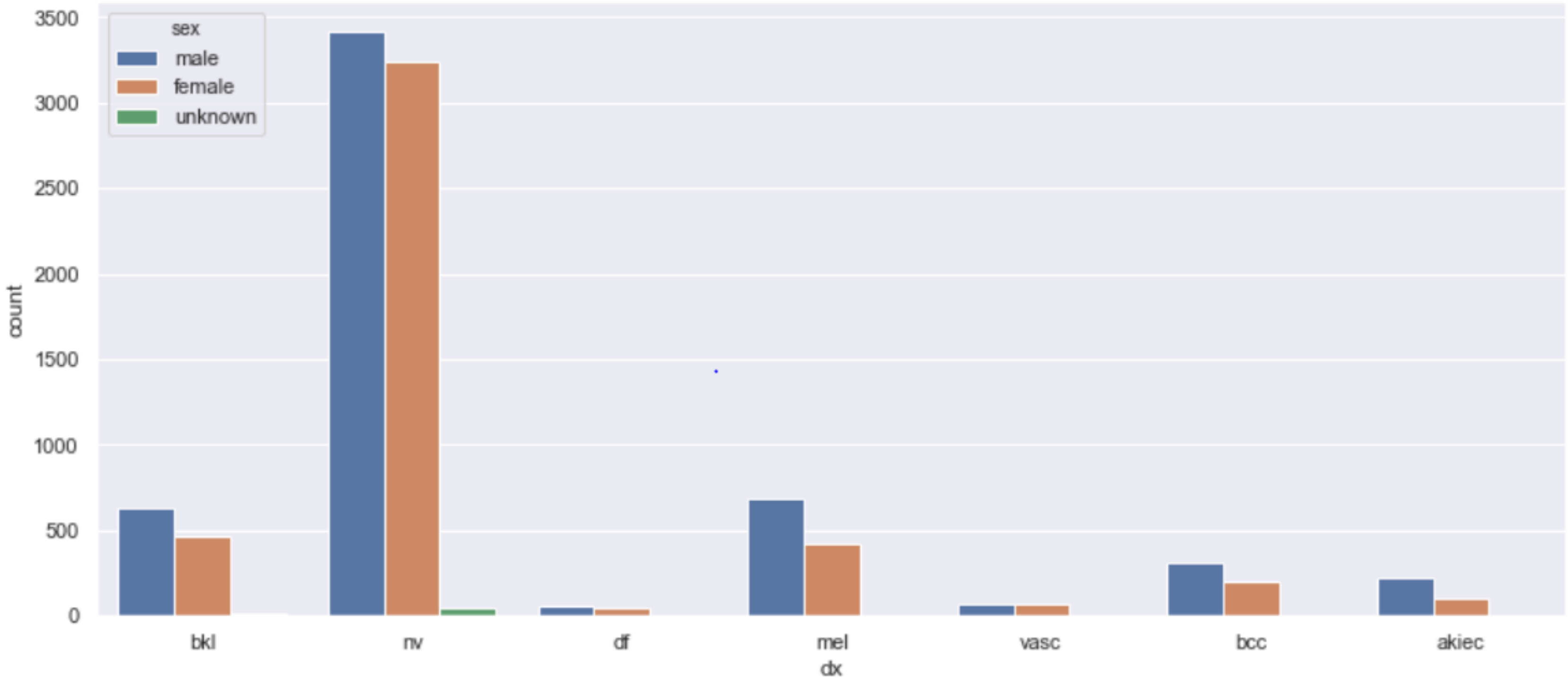
Exploratory Data Analysis - Correlation between Localization and Age



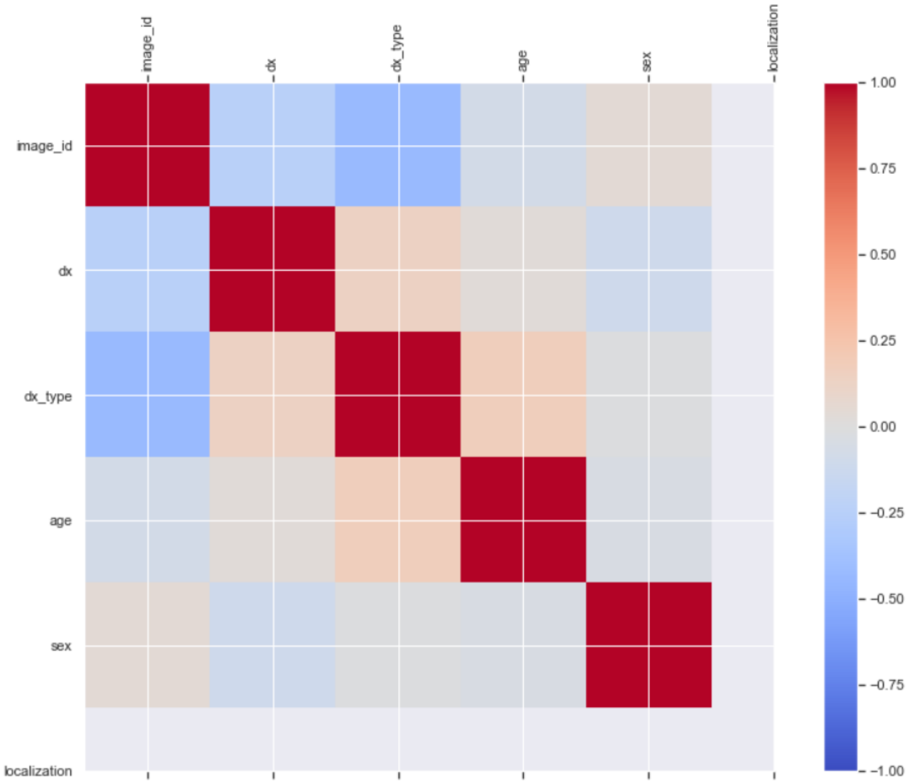
Exploratory Data Analysis - Correlation between Age and Type of Cancer



Exploratory Data Analysis - Correlation between Type of Cancer and Sex



Exploratory Data Analysis - Correlation between parameters

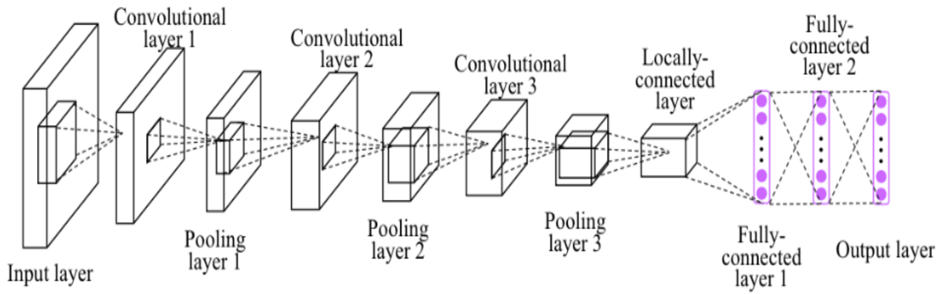


Data Preprocessing

- Remove the Null Values
- Remove the Duplicates
- Splitting the dataset into Training and Testing sets
- Convert categorical columns into Numerical columns using One Hot Encoding and Label Encoding
- Resizing the Images
- Normalization

CNN Architecture

1. Convolution layer -Conv2D
2. Pooling layer MaxPooling2D
3. Flatten layer
4. Fully connected layer -Dense



CNN Architecture

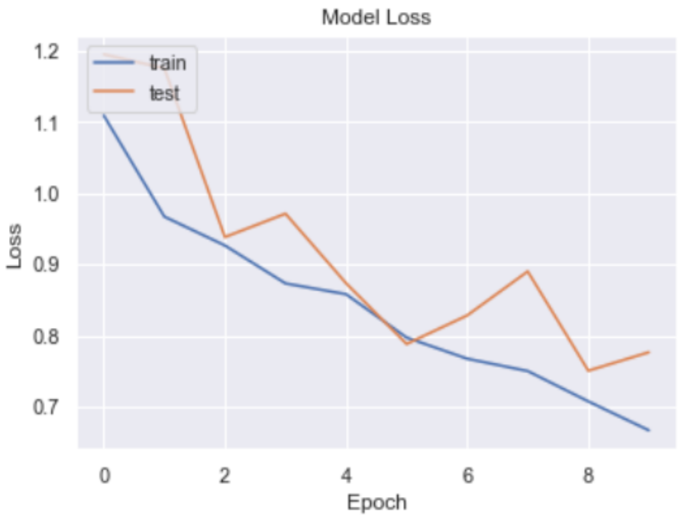
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=(100,100,3),padding='same'))
model.add(MaxPooling2D((2, 2),padding='same'))
model.add(Dropout(0.20))

model.add(Conv2D(64, (3, 3), activation='relu',padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
model.add(Dropout(0.40))

model.add(Conv2D(128, (3, 3), activation='relu',padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
model.add(Dropout(0.20))

model.add(Flatten())

model.add(Dense(64, activation='linear'))
model.add(LeakyReLU(alpha=0.1))
model.add(Dense(128, activation='linear'))
model.add(Dense(256, activation='linear'))
model.add(Dense(7, activation='softmax'))
model.summary()
```

Setting Optimizer and Fitting the model

```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
history=model.fit(train_X,one_hot_train,batch_size=128,epochs=10,validation_split=0.2)
```

```
Train on 6409 samples, validate on 1603 samples
Epoch 1/10
6409/6409 [==============================] - 122s 19ms/step - loss: 1.1092 - accuracy: 0.6653 - val_loss: 1.1953 - val_accurac
y: 0.6569
Epoch 2/10
6409/6409 [==============================] - 112s 17ms/step - loss: 0.9671 - accuracy: 0.6753 - val_loss: 1.1755 - val_accurac
y: 0.6438
Epoch 3/10
6409/6409 [==============================] - 112s 18ms/step - loss: 0.9265 - accuracy: 0.6706 - val_loss: 0.9383 - val_accurac
y: 0.6644
Epoch 4/10
6409/6409 [==============================] - 111s 17ms/step - loss: 0.8731 - accuracy: 0.6878 - val_loss: 0.9712 - val_accurac
y: 0.6837
Epoch 5/10
6409/6409 [==============================] - 111s 17ms/step - loss: 0.8580 - accuracy: 0.6914 - val_loss: 0.8735 - val_accurac
y: 0.6893
Epoch 6/10
6409/6409 [==============================] - 111s 17ms/step - loss: 0.7970 - accuracy: 0.7076 - val_loss: 0.7880 - val_accurac
y: 0.7137
Epoch 7/10
6409/6409 [==============================] - 113s 18ms/step - loss: 0.7673 - accuracy: 0.7202 - val_loss: 0.8282 - val_accurac
y: 0.6993
Epoch 8/10
6409/6409 [==============================] - 111s 17ms/step - loss: 0.7500 - accuracy: 0.7190 - val_loss: 0.8900 - val_accurac
y: 0.6949
Epoch 9/10
6409/6409 [==============================] - 110s 17ms/step - loss: 0.7073 - accuracy: 0.7327 - val_loss: 0.7504 - val_accurac
y: 0.7118
Epoch 10/10
6409/6409 [==============================] - 110s 17ms/step - loss: 0.6666 - accuracy: 0.7510 - val_loss: 0.7764 - val_accurac
y: 0.7255
```

Model Evaluation



Conclusion

Accuracy is higher if model is trained on more samples of lower resolution than small samples of high resolutions.

We have achieved the accuracy of 75%.

Future Scope

Going forward, we can continue to refine the model to achieve a stable decrease in loss function with every epoch, build an interface such that given an image of a skin lesion within the two classes, the output will give a % probability of which of the seven classes it belongs to.

Q & A

Thank You