**Lipid-Starch Identification**

**Business Value:** Reduce the manual effort of finding the percentage of lipid and starch in the algae cells for the microscopic images using the ImageJ tool.

The application automatically predicts the lipid, starch and the cell classes by leveraging the image segmentation technique using deep learning for the microscopic images and then find the percentages of different classes.

Scope of the work for Phase 1:

Initial sample set of 246 images

Conversion into JPEG and resizing the images

Image segmentation and extracting the percentage value

**Approach:** The following steps are followed:

1. Conversion of .tif image format to .jpeg image format and removed some images which are not labelled correctly.
2. Data Preparation: Generated the masked images for the 246 images using the Python script.
3. Data Preprocessing: Label Encoding for the 246 images and the random train-test split of 90-10. 25 images considered as test images and rest of the images are for model training.
4. Algorithm used: Random Forest with and without CNN features and VGG16 Imagenet weights, XGBoost, Simple Unet, Unet with VGG16, ResNet and Inception architectures, Ensemble learning of Unet with VGG16, ResNet and Inception architectures.

**Brief explanation about the used algorithms:**

As the original images have different dimensions, so resized the image as 128\*128. Total classes are 4 as Lipid, Starch,Cell and background in the images. As all the images have taken in arrays and a particular image has 3 channels. So multi-dimensional arrays need to be flatten, encode and reshape.

**Random Forest:** Used 30 de-correlated trees with bootstrapped samples and with entropy as an impurity measure to find the maximum information gain and minimize the variance to overcome the overfitting problem. Also, used the Gabor filters like canny, sober edge detector, texture detection etc for the feature extraction.

**XGBoost:** Used the Boosting principle as training the classifiers sequentially and each classifier is trained based on the performance of the previous trained classifier. Final classifier is the weighted sum of the component classifiers.

**Simple Unet:** It is a simple encoder-decoder based deep learning architecture with skip connections. For the encoder side, convolution, max-pooling and the dropout layers are used and 128\*128 resized image is compressed into 8\*8 dimensions with 256 features and then decoding is done to produce 4 classes with different probabilities using the softmax layer.

**Unet with ResNet, VGG16 and Inception:** Resnet, VGG16 AND Inception are considered as the backbone and it is the model to be used for the encoder part of the UNet. This lets us benefit from the transfer learning by using pretrained weights like imagenet etc. For model training 50 epochs, batch size=8 are used with learning rate as 0.0001 and with softmax as the activation function.

**Improved Unet with Ensemble Learning:**

3 backbones have been used as **resnet34, inceptionv3 and vgg16.** For all these 3 backbones, the test images are pre-processed and then saved model is loaded and to make ensemble, grid search is used to find the best weight combination to find the best model that maximize the mean IoU score.

**Solution:** Mean IoU is considered as a performance metric and Focal loss is considered as a loss function.

**Model Performance:**

1. Simple Unet: 62.00% mean IoU
2. Unet(VGG16): 88.60% mean IoU
3. Unet(ResNet): 90.40% mean IoU
4. Unet(Inception): 93.60% mean IoU
5. Unet(Ensemble): 94.00% mean IoU

**Next Step:**

Model Refinement and Scaling:

More the data, better the model training. Building more generic robust model by integrating more data points.

Building User Interface and Model Deployment.

**Data Issues:**

Lack of sufficient number of data points for model training (typically model training for image segmentation with deep learning happens on minimum tens of thousands of data)