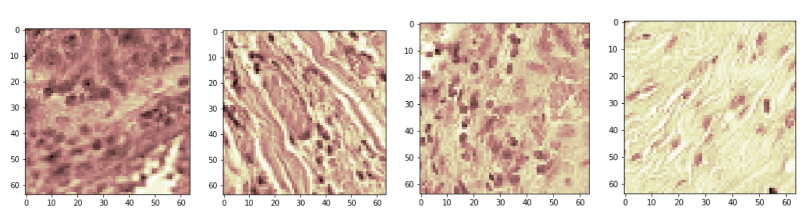
Histopathologic Cancer Detection

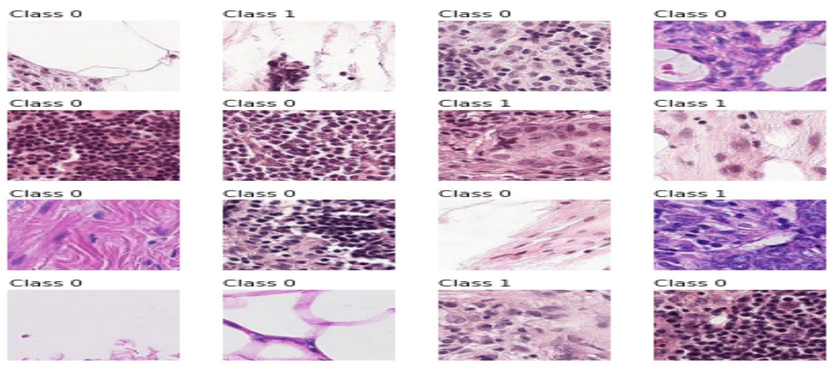
Abstract:
Cancer has emerged as a significant concern and hazard to people all over the world, and it is now the leading cause of death in almost every country. we have used various models to predict it using various pathologic tumor images.

Introduction:
There are three basic phases involved in our project implementation:
1)Pre-processing
2)Feature extraction
3)Classification

Pre-processing

The raw data collected by sensors is turned into a structure during the pre-processing step, from which the most important aspects related to the domain are identified for further analysis. This main purpose is to remove background noise from the input image in order to improve its overall quality. During the photographic and slide processing processes, a large number of differences may occur. The brightness of the backdrop in the resulting histopathological image is not always consistent with the foreground since the image is compressed. During the photographic and slide processing processes, a large number of differences may occur.Throughout the photographing and slide processing procedure, a large number of differences may occur. The brightness of the backdrop in the resulting histopathological image is not always consistent with the foreground because the image is acquired in a compressed format.Using the gray format the histopathological image that was acquired are shown below:





The above image contains labels in accordance with their images.

FEATURE EXTRACTION

>>

The underlying premise of feature extraction is that by reducing the size of the feature space, all the calculations may be completed in the shortest amount of time. Most of the time, a feature extraction technique is divided into three key parts. The steps are as follows: the construction of possible features using various transformation techniques, the selection of the optimal . features from among the possible features, which results in improved system performance, and finally, the matching of the features using various classifiers for recognizing the objects.

>>

Manual feature extraction requires identifying and describing the features that are relevant for a given problem and implementing a way to extract those features. In many situations, having a good understanding of the background or domain can help make informed decisions as to which features could be useful

DATASET:

There are a total of 220025 images available to us which we have divided into training and testing data.

Methodology:

We have used following models:

1)CNN

2)LBGM

3)XGBOOST

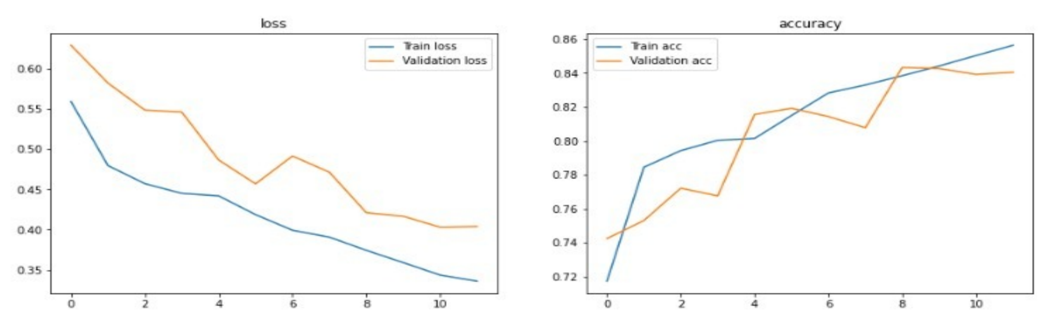
Implementation of models:

1)CNN:A convolutional neural network (CNN) is a type of artificial neural network_used in image recognition and processing that is specifically designed to process pixel data.we have used his model cause it gives better results on image classification.

Experimental analysis:

cnn1:

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_26 (Conv2D)          (None, 64, 64, 32)        896

 conv2d_27 (Conv2D)          (None, 64, 64, 32)        9248

 conv2d_28 (Conv2D)          (None, 64, 64, 32)        9248

 dropout_8 (Dropout)         (None, 64, 64, 32)        0

 max_pooling2d_10 (MaxPoolin  (None, 21, 21, 32)       0
 g2D)

 conv2d_29 (Conv2D)          (None, 21, 21, 64)        18496

 conv2d_30 (Conv2D)          (None, 21, 21, 64)        36928

 conv2d_31 (Conv2D)          (None, 21, 21, 64)        36928

 dropout_9 (Dropout)         (None, 21, 21, 64)        0

 max_pooling2d_11 (MaxPoolin  (None, 7, 7, 64)         0
 g2D)

 conv2d_32 (Conv2D)          (None, 7, 7, 128)         73856

 conv2d_33 (Conv2D)          (None, 7, 7, 128)         147584

 conv2d_34 (Conv2D)          (None, 7, 7, 128)         147584

 dropout_10 (Dropout)        (None, 7, 7, 128)         0

 max_pooling2d_12 (MaxPoolin  (None, 2, 2, 128)        0
 g2D)

 flatten_4 (Flatten)         (None, 512)               0

 dense_10 (Dense)            (None, 128)               65664

 dropout_11 (Dropout)        (None, 128)               0

 dense_11 (Dense)            (None, 1)                 129

=================================================================
Total params: 546,561
Trainable params: 546,561
Non-trainable params: 0
_________________________________________________________________
```



The image below shows loss function and accuracy score of training and testing data.

Output and accuracy:

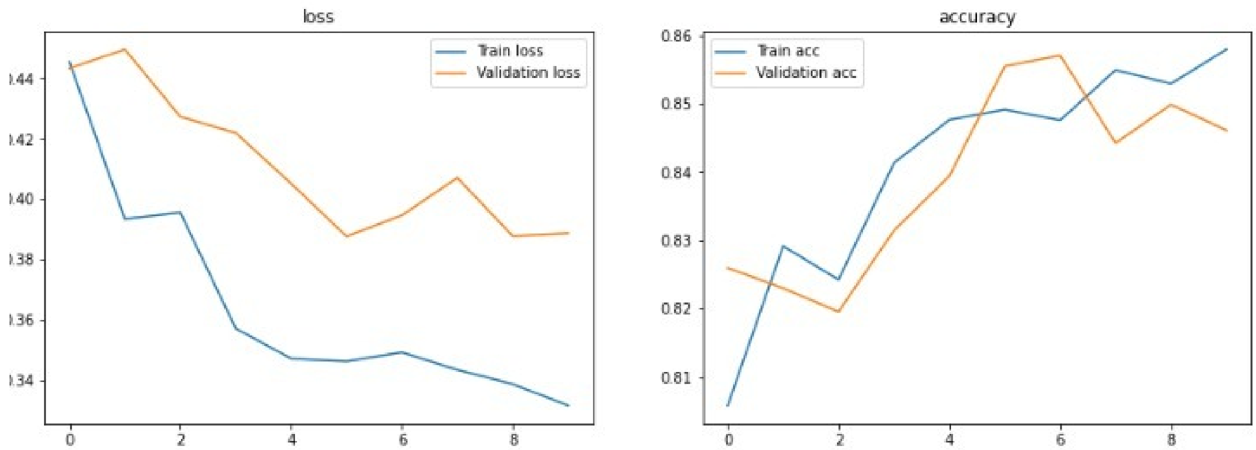
```
Epoch 1/15
100/100 [==============================] - 452s 5s/step - loss: 0.5592 - accuracy: 0.7174 - val_loss: 0.6293 - val_accuracy: 0.7425 - lr: 0.0010
Epoch 2/15
100/100 [==============================] - 483s 5s/step - loss: 0.4797 - accuracy: 0.7845 - val_loss: 0.5824 - val_accuracy: 0.7531 - lr: 0.0010
Epoch 3/15
100/100 [==============================] - 430s 4s/step - loss: 0.4572 - accuracy: 0.7943 - val_loss: 0.5487 - val_accuracy: 0.7722 - lr: 0.0010
Epoch 4/15
100/100 [==============================] - 429s 4s/step - loss: 0.4451 - accuracy: 0.8004 - val_loss: 0.5462 - val_accuracy: 0.7677 - lr: 0.0010
Epoch 5/15
100/100 [==============================] - 428s 4s/step - loss: 0.4420 - accuracy: 0.8015 - val_loss: 0.4866 - val_accuracy: 0.8156 - lr: 0.0010
Epoch 6/15
100/100 [==============================] - 435s 4s/step - loss: 0.4186 - accuracy: 0.8150 - val_loss: 0.4569 - val_accuracy: 0.8192 - lr: 0.0010
Epoch 7/15
100/100 [==============================] - 446s 4s/step - loss: 0.3992 - accuracy: 0.8283 - val_loss: 0.4916 - val_accuracy: 0.8144 - lr: 0.0010
Epoch 8/15
100/100 [==============================] - ETA: 0s - loss: 0.3906 - accuracy: 0.8330
Epoch 8: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
100/100 [==============================] - 466s 5s/step - loss: 0.3906 - accuracy: 0.8330 - val_loss: 0.4714 - val_accuracy: 0.8078 - lr: 0.0010
Epoch 9/15
100/100 [==============================] - 441s 4s/step - loss: 0.3742 - accuracy: 0.8384 - val_loss: 0.4211 - val_accuracy: 0.8433 - lr: 1.0000e-04
Epoch 10/15
100/100 [==============================] - 431s 4s/step - loss: 0.3590 - accuracy: 0.8441 - val_loss: 0.4166 - val_accuracy: 0.8427 - lr: 1.0000e-04
Epoch 11/15
100/100 [==============================] - ETA: 0s - loss: 0.3434 - accuracy: 0.8503
Epoch 11: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
100/100 [==============================] - 441s 4s/step - loss: 0.3434 - accuracy: 0.8503 - val_loss: 0.4030 - val_accuracy: 0.8392 - lr: 1.0000e-04
Epoch 12/15
100/100 [==============================] - ETA: 0s - loss: 0.3359 - accuracy: 0.8563Restoring model weights from the end of the best epoch: 9.
100/100 [==============================] - 479s 5s/step - loss: 0.3359 - accuracy: 0.8563 - val_loss: 0.4038 - val_accuracy: 0.8405 - lr: 1.0000e-05
Epoch 12: early stopping
```

CNN2:

Layer (type)	Output Shape	Param #
conv2d_22 (Conv2D)	(None, 94, 94, 64)	1792
batch_normalization_6 (BatchNormalization)	(None, 94, 94, 64)	256
conv2d_23 (Conv2D)	(None, 92, 92, 128)	73856
batch_normalization_7 (BatchNormalization)	(None, 92, 92, 128)	512
max_pooling2d_8 (MaxPooling2D)	(None, 30, 30, 128)	0
conv2d_24 (Conv2D)	(None, 28, 28, 256)	295168
batch_normalization_8 (BatchNormalization)	(None, 28, 28, 256)	1024
max_pooling2d_9 (MaxPooling2D)	(None, 9, 9, 256)	0
conv2d_25 (Conv2D)	(None, 7, 7, 512)	1180160
batch_normalization_9 (BatchNormalization)	(None, 7, 7, 512)	2048
global_max_pooling2d_1 (GlobalMaxPooling2D)	(None, 512)	0
flatten_3 (Flatten)	(None, 512)	0
dense_7 (Dense)	(None, 512)	262656
batch_normalization_10 (BatchNormalization)	(None, 512)	2048
dense_8 (Dense)	(None, 1024)	525312
batch_normalization_11 (BatchNormalization)	(None, 1024)	4096
dense_9 (Dense)	(None, 1)	1025

Total params: 2,349,953
Trainable params: 2,344,961
Non-trainable params: 4,992

The image below shows loss function and accuracy score of training and testing data.



OUTPUT AND ACCURACY:

```
Epoch 1/10
70/70 [==============================] - 390s 6s/step - loss: 0.4454 - accuracy: 0.8058 - val_loss: 0.4433 - val_accuracy: 0.8259 - lr: 0.0010
Epoch 2/10
70/70 [==============================] - 313s 4s/step - loss: 0.3934 - accuracy: 0.8291 - val_loss: 0.4496 - val_accuracy: 0.8230 - lr: 0.0010
Epoch 3/10
70/70 [==============================] - ETA: 0s - loss: 0.3956 - accuracy: 0.8242
Epoch 3: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
70/70 [==============================] - 313s 4s/step - loss: 0.3956 - accuracy: 0.8242 - val_loss: 0.4273 - val_accuracy: 0.8195 - lr: 0.0010
Epoch 4/10
70/70 [==============================] - 347s 5s/step - loss: 0.3572 - accuracy: 0.8413 - val_loss: 0.4220 - val_accuracy: 0.8314 - lr: 1.0000e-04
Epoch 5/10
70/70 [==============================] - 311s 4s/step - loss: 0.3472 - accuracy: 0.8476 - val_loss: 0.4052 - val_accuracy: 0.8394 - lr: 1.0000e-04
Epoch 6/10
70/70 [==============================] - 314s 4s/step - loss: 0.3463 - accuracy: 0.8491 - val_loss: 0.3877 - val_accuracy: 0.8555 - lr: 1.0000e-04
Epoch 7/10
70/70 [==============================] - 315s 5s/step - loss: 0.3492 - accuracy: 0.8475 - val_loss: 0.3947 - val_accuracy: 0.8570 - lr: 1.0000e-04
Epoch 8/10
70/70 [==============================] - 315s 5s/step - loss: 0.3435 - accuracy: 0.8549 - val_loss: 0.4071 - val_accuracy: 0.8442 - lr: 1.0000e-04
Epoch 9/10
70/70 [==============================] - ETA: 0s - loss: 0.3388 - accuracy: 0.8529
Epoch 9: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
70/70 [==============================] - 350s 5s/step - loss: 0.3388 - accuracy: 0.8529 - val_loss: 0.3878 - val_accuracy: 0.8498 - lr: 1.0000e-04
Epoch 10/10
70/70 [==============================] - ETA: 0s - loss: 0.3317 - accuracy: 0.8579Restoring model weights from the end of the best epoch: 7.
70/70 [==============================] - 314s 4s/step - loss: 0.3317 - accuracy: 0.8579 - val_loss: 0.3887 - val_accuracy: 0.8461 - lr: 1.0000e-05
Epoch 10: early stopping
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_22 (Conv2D)          (None, 94, 94, 64)        1792

 batch_normalization_6 (Batc  (None, 94, 94, 64)       256
 hNormalization)

 conv2d_23 (Conv2D)          (None, 92, 92, 128)       73856

 batch_normalization_7 (Batc  (None, 92, 92, 128)      512
 hNormalization)

 max_pooling2d_8 (MaxPooling  (None, 30, 30, 128)      0
 2D)

 conv2d_24 (Conv2D)          (None, 28, 28, 256)       295168

 batch_normalization_8 (Batc  (None, 28, 28, 256)      1024
 hNormalization)

 max_pooling2d_9 (MaxPooling  (None, 9, 9, 256)        0
 2D)

 conv2d_25 (Conv2D)          (None, 7, 7, 512)         1180160

 batch_normalization_9 (Batc  (None, 7, 7, 512)        2048
 hNormalization)

 global_max_pooling2d_1 (Glo  (None, 512)              0
 balMaxPooling2D)

 flatten_3 (Flatten)         (None, 512)               0

 dense_7 (Dense)             (None, 512)               262656

 batch_normalization_10 (Bat  (None, 512)              2048
 chNormalization)

 dense_8 (Dense)             (None, 1024)              525312

 batch_normalization_11 (Bat  (None, 1024)             4096
 chNormalization)

 dense_9 (Dense)             (None, 1)                 1025

=================================================================
Total params: 2,349,953
Trainable params: 2,344,961
Non-trainable params: 4,992
_________________________________________________________________
```

XGBOOST:

XGBoost is a powerful approach for building supervised regression models. The validity of this statement can be inferred by knowing about its (XGBoost) objective function and base learners.

Parameters:

```
The best verbosity is: 0
The best depth: 4
Best n_estimators val is: 1000
```

Output and accuracy:

```
Training Accuracy: 0.969895613428
Test Accuracy: 0.908608848721
```

LBGM:

PARAMETERS:

```
The best verbosity is: 1
The best depth is: 7
Best n_estimators val is: 2000
```

ACCURACY:

```
Training Accuracy: 0.9666666666666667
Test accuracy: 0.92471156565657
```

TABLE

SERIAL NO.	MODEL	ACCURACY
1.	CNN1	85.63
2.	CNN2	85.79
3.	XGBOOST	90.86
4.	LGBM	92.47

REFERENCES:

- Caruana, Rich, et al. “Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission.” Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. (2015).
- Fisher, Aaron, Cynthia Rudin, and Francesca Dominici. “All models are wrong, but many are useful: Learning a variable’s importance by studying an entire class of prediction models simultaneously. (2018).
- https://en.wikipedia.org/wiki/Convolutional_neural_network