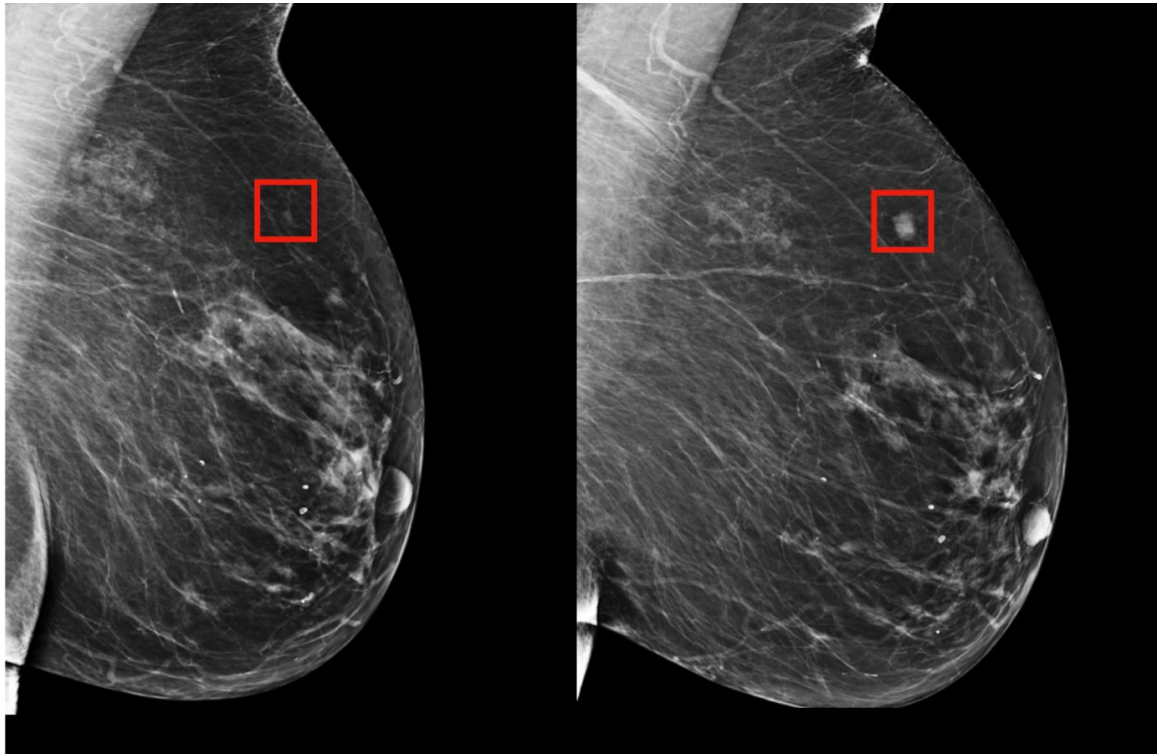
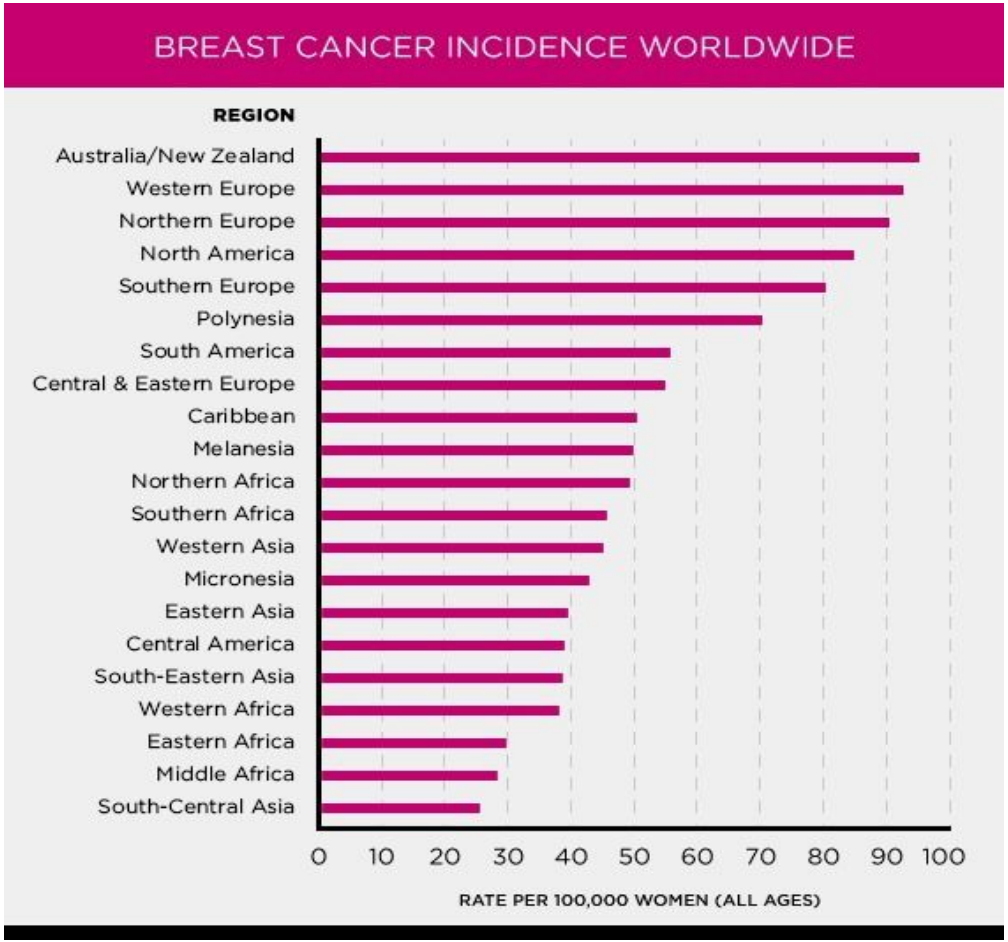
Detecting Breast Cancer with Data Mining Techniques

Team 11:
Meghna Bajoria [016661528],
Shakshi Richhariya[016043105],
Vinit Kanani[016651323],
Kapil Wanaskar[016649880],



Reference

Shakshi

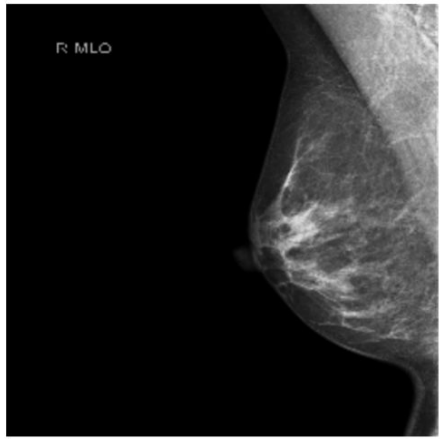


https://www.komen.org/breast-cancer/facts-statistics/breast-cancer-statistics/

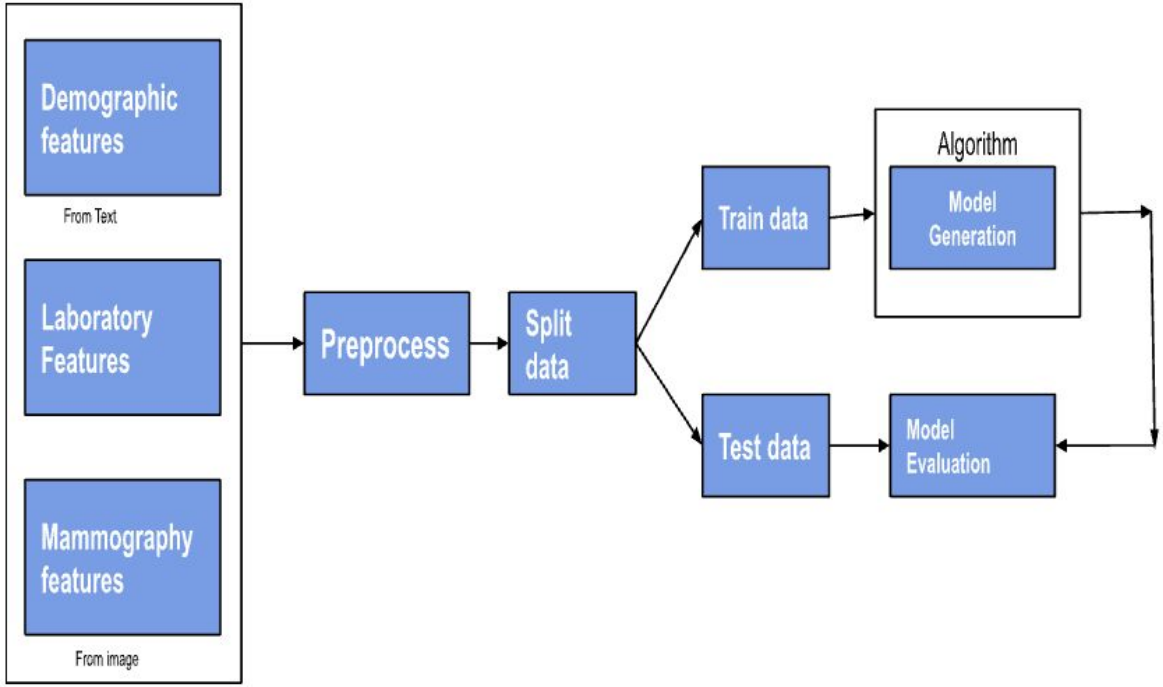
The goal: is to identify cases of breast cancer in mammograms from screening exams.

It is important to identify cases of cancer for obvious reasons, but false positives also have downsides for patients. As millions of women get mammograms each year, a useful machine learning tool could help a great many people.

```
from IPython.display import Image
Image('/content/10011_1031443799.png')
```



Flow Diagram from Data Extraction to Model Training and Testing



Flow

1. Data Extraction
2. EDA (Exploratory data analysis (EDA))
3. Data pre processing (handling missing values, outliers, Normalization)
4. Model training and evaluation (metrics, such as accuracy, precision, recall, and F1-score)
5. Front-end and backend deployment

```
[ ] import pandas as pd
    # Read the CSV file
    df0 = pd.read_csv('train.csv')
    # source: https://www.kaggle.com/competitions/rsna-breast-cancer-detection/data?select=train.csv
    df0
```

	site_id	patient_id	image_id	laterality	view	age	cancer	biopsy	invasive	BIRADS	implant	density	machine_id	difficult_negative_case
0	2	10006	462822612	L	CC	61.0	0	0	0	NaN	0	NaN	29	False
1	2	10006	1459541791	L	MLO	61.0	0	0	0	NaN	0	NaN	29	False
2	2	10006	1864590858	R	MLO	61.0	0	0	0	NaN	0	NaN	29	False
3	2	10006	1874946579	R	CC	61.0	0	0	0	NaN	0	NaN	29	False
4	2	10011	220375232	L	CC	55.0	0	0	0	0.0	0	NaN	21	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
54701	1	9973	1729524723	R	MLO	43.0	0	0	0	1.0	0	C	49	False
54702	1	9989	63473691	L	MLO	60.0	0	0	0	NaN	0	C	216	False
54703	1	9989	1078943060	L	CC	60.0	0	0	0	NaN	0	C	216	False
54704	1	9989	398038886	R	MLO	60.0	0	0	0	0.0	0	C	216	True
54705	1	9989	439796429	R	CC	60.0	0	0	0	0.0	0	C	216	True

54706 rows × 14 columns

[train/test].csv Metadata for each patient and image. Only the first few rows of the test set are available for download.

- `site_id` - ID code for the source hospital.
- `patient_id` - ID code for the patient.
- `image_id` - ID code for the image.
- `laterality` - Whether the image is of the left or right breast.
- `view` - The orientation of the image. The default for a screening exam is to capture two views per breast.
- `age` - The patient's age in years.
- `implant` - Whether or not the patient had breast implants. Site 1 only provides breast implant information at the patient level, not at the breast level.
- `density` - A rating for how dense the breast tissue is, with A being the least dense and D being the most dense. Extremely dense tissue can make diagnosis more difficult. Only provided for train.
- `machine_id` - An ID code for the imaging device.
- `cancer` - Whether or not the breast was positive for malignant cancer. The target value. Only provided for train.
- `biopsy` - Whether or not a follow-up biopsy was performed on the breast. Only provided for train.
- `invasive` - If the breast is positive for cancer, whether or not the cancer proved to be invasive. Only provided for train.
- `BIRADS` - 0 if the breast required follow-up, 1 if the breast was rated as negative for cancer, and 2 if the breast was rated as normal. Only provided for train.
- `prediction_id` - The ID for the matching submission row. Multiple images will share the same prediction ID. Test only.
- `difficult_negative_case` - True if the case was unusually difficult. Only provided for train.

```
[ ] # Move the kaggle.json file to the correct location and set the appropriate permissions:
    # FOR GOOGLE COLLAB
    !mkdir -p ~/.kaggle
    !mv kaggle.json ~/.kaggle/
    !chmod 600 ~/.kaggle/kaggle.json
```

```
# Make sure to use the dataset's API command found on the dataset page under the "Three Dots" menu > "Copy API command":

# source: https://www.kaggle.com/datasets/theoviel/rsna-breast-cancer-256-pngs
!kaggle datasets download -d theoviel/rsna-breast-cancer-256-pngs #[1 GB]

# source: https://www.kaggle.com/datasets/theoviel/rsna-breast-cancer-512-pngs
# !kaggle datasets download -d theoviel/rsna-breast-cancer-512-pngs #[4 GB]

# source: https://www.kaggle.com/datasets/theoviel/rsna-breast-cancer-1024-pngs
# !kaggle datasets download -d theoviel/rsna-breast-cancer-1024-pngs #[14 GB]
```

```
Downloading rsna-breast-cancer-256-pngs.zip to /content
100% 995M/998M [00:11<00:00, 80.3MB/s]
100% 998M/998M [00:11<00:00, 87.5MB/s]
```

```
[ ] # Unzip the downloaded dataset:
    !unzip rsna-breast-cancer-256-pngs.zip
```

```
Streaming output truncated to the last 5000 lines.
  inflating: 63617_159110654.png
  inflating: 63617_1855699072.png
  inflating: 63617_1917877119.png
  inflating: 63617_887426674.png
  inflating: 63623_1186739164.png
  inflating: 63623_1657034655.png
  inflating: 63623_335778602.png
  inflating: 63623_625719002.png
  inflating: 63626_1154699412.png
  inflating: 63626_549972940.png
```

When working with images in a machine learning task like identifying cases of breast cancer in mammograms, there are a variety of features that can be extracted from the images. Here are some common types of features that can be extracted:

[1] Pixel-based features: This involves extracting features directly from the pixels of the images, such as color intensity, texture, and shape.

[2] Statistical features: This involves calculating statistical properties of the pixel values in the images, such as mean, variance, and skewness.

[3] Structural features: This involves analyzing the structure of the images, such as identifying edges, shapes, and patterns.

[4] Domain-specific features: This involves extracting features that are specific to the domain of breast cancer detection, such as the presence of masses, calcifications, or architectural distortions.

[5] Deep learning features: This involves using pre-trained convolutional neural networks (CNNs) to extract features from the images, which can be used as input to a machine learning model.

Some specific examples of features that could be extracted from mammogram images include:

[1] Mean and standard deviation of pixel values in the image

[2] Histogram of pixel values in the image

[3] Texture features, such as gray-level co-occurrence matrix (GLCM) features or gray-level run length matrix (GLRLM) features

[4] Shape features, such as the circularity or compactness of masses or the branching structure of ductal carcinoma in situ (DCIS)

[5] Features related to the distribution and density of microcalcifications, such as their size, shape, and clustering

[6] Deep learning features extracted from pre-trained CNNs such as VGG, ResNet, or Inception models.

Ultimately, the specific features that are most useful for identifying breast cancer in mammograms will depend on the particular dataset and the goals of the machine learning task. It is important to carefully analyze the images and experiment with different feature extraction methods to determine which features are most predictive of cancer.

Kapil

```
[ ] # Move the kaggle.json file to the correct location and set the appropriate permissions:
    # FOR GOOGLE COLLAB
    !mkdir -p ~/.kaggle
    !mv kaggle.json ~/.kaggle/
    !chmod 600 ~/.kaggle/kaggle.json
```

```
# Make sure to use the dataset's API command found on the dataset page under the "Three Dots" menu > "Copy API command":

# source: https://www.kaggle.com/datasets/theoviel/rsna-breast-cancer-256-pngs
!kaggle datasets download -d theoviel/rsna-breast-cancer-256-pngs #[1 GB]

# source: https://www.kaggle.com/datasets/theoviel/rsna-breast-cancer-512-pngs
# !kaggle datasets download -d theoviel/rsna-breast-cancer-512-pngs #[4 GB]

# source: https://www.kaggle.com/datasets/theoviel/rsna-breast-cancer-1024-pngs
# !kaggle datasets download -d theoviel/rsna-breast-cancer-1024-pngs #[14 GB]
```

```
Downloading rsna-breast-cancer-256-pngs.zip to /content
100% 995M/998M [00:11<00:00, 80.3MB/s]
100% 998M/998M [00:11<00:00, 87.5MB/s]
```

```
[ ] # Unzip the downloaded dataset:
    !unzip rsna-breast-cancer-256-pngs.zip
```

```
Streaming output truncated to the last 5000 lines.
  inflating: 63617_159110654.png
  inflating: 63617_1855699072.png
  inflating: 63617_1917877119.png
  inflating: 63617_887426674.png
  inflating: 63623_1186739164.png
  inflating: 63623_1657034655.png
  inflating: 63623_335778602.png
  inflating: 63623_625719002.png
  inflating: 63626_1154699412.png
  inflating: 63626_549972940.png
```

Creativity : Data Extraction from Images

```python
# Function to calculate GLCM features
def glcm_features(gray_image):
    glcm = greycomatrix(gray_image, [1], [0],  symmetric=True, normed=True)
    contrast = greycoprops(glcm, 'contrast')[0, 0]
    dissimilarity = greycoprops(glcm, 'dissimilarity')[0, 0]
    homogeneity = greycoprops(glcm, 'homogeneity')[0, 0]
    energy = greycoprops(glcm, 'energy')[0, 0]
    correlation = greycoprops(glcm, 'correlation')[0, 0]
    return [contrast, dissimilarity, homogeneity, energy, correlation]
```

Creativity : Data Extraction from Images

```
# Function to calculate shape features (circularity and compactness)
def shape_features(img):
    # Apply binary threshold
    thresh = threshold_otsu(img)
    binary_img = img > thresh

    # Label the binary image
    labeled_img = label(binary_img)
    properties = regionprops(labeled_img)

    if not properties:
        return [None, None]

    area = properties[0].area
    perimeter = properties[0].perimeter
    compactness = (perimeter**2) / (4 * np.pi * area) if area > 0 and perimeter > 0 else None
    circularity = 4 * np.pi * area / (perimeter**2) if area > 0 and perimeter > 0 else None

    return [compactness, circularity]
```

Creativity : Data Extraction from Images

```
# Function to extract microcalcification features (size, shape, and clustering)
def microcalcification_features(img):
    binary_img = closing(img > np.mean(img), square(3))
    labeled_img = label(binary_img)
    remove_small_objects(labeled_img, 10, in_place=True)

    properties = regionprops(labeled_img)
    areas = [prop.area for prop in properties]
    solidity = [prop.solidity for prop in properties]

    if not areas:
        return [None, None, None]

    mean_size = np.mean(areas)
    mean_solidity = np.mean(solidity)
    clustering = len(areas) / np.sum(areas)

    return [mean_size, mean_solidity, clustering]
```

Creativity : Data Extraction from Images

```
# Initialize an empty list to store image features
image_features = []

# Iterate through images in the /content folder
images_folder = "/content"  # Replace this with the path to your folder containing the image files
# image_count = 0

for file_name in os.listdir(images_folder):
  # if image_count >= 15000:
  #       break
  # for file_name in os.listdir('/content'):
  if file_name.endswith('.png'):
      # Read the image
      img = cv2.imread(os.path.join(images_folder, file_name))

      # Convert image to grayscale
      gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

      # Calculate color intensity
      color_intensity = np.mean(img, axis=(0, 1))

      # Calculate variance
      variance = np.var(gray_img)

      # Calculate skewness
      skewness = skew(gray_img.flatten())

      # Calculate edges
      edges = canny(gray_img)
```

Creativity : Data Extraction from Images

```
df
```

	file_name	color_intensity_r	color_intensity_g	color_intensity_b	mean_pixel_value	std_pixel_value	variance	skewness	contrast	dissimil
0	24231_1599132094.png	21.669540	21.669540	21.669540	21.669540	41.618134	1732.069065	2.223763	186.391054	4.7
1	10838_591123709.png	80.702255	80.702255	80.702255	80.702255	69.249010	4795.425401	-0.185761	161.732475	3.3
2	27667_830917739.png	45.809540	45.809540	45.809540	45.809540	54.380373	2957.224955	1.319218	279.208655	7.9
3	16339_1765002339.png	21.175934	21.175934	21.175934	21.175934	47.596471	2265.424064	1.864654	31.575797	1.2
4	2606_72547626.png	21.496078	21.496078	21.496078	21.496078	40.860531	1669.583023	2.136658	234.588542	5.2
...	...	...	...	...	...	...	...	...	...	
54701	57444_2114947064.png	30.386978	30.386978	30.386978	30.386978	50.009012	2500.901319	1.742150	449.614308	8.5
54702	1878_189362032.png	35.229370	35.229370	35.229370	35.229370	39.112316	1529.773256	1.029031	325.719899	8.1
54703	45339_1546789692.png	43.123657	43.123657	43.123657	43.123657	74.371804	5531.165159	1.178659	30.847656	1.4
54704	23855_1024724669.png	24.096588	24.096588	24.096588	24.096588	37.214050	1384.885507	1.505538	180.844393	4.9
54705	3525_1032766690.png	71.472580	71.472580	71.472580	71.472580	75.002949	5625.442424	0.199063	64.805438	3.0

54706 rows × 19 columns

```
[ ] import pandas as pd
    # Read the CSV file
    df0 = pd.read_csv('train.csv')
    # source: https://www.kaggle.com/competitions/rsna-breast-cancer-detection/data?select=train.csv
    df0
```

	site_id	patient_id	image_id	laterality	view	age	cancer	biopsy	invasive	BIRADS	implant	density	machine_id	difficult_negative_case
0	2	10006	462822612	L	CC	61.0	0	0	0	NaN	0	NaN	29	False
1	2	10006	1459541791	L	MLO	61.0	0	0	0	NaN	0	NaN	29	False
2	2	10006	1864590858	R	MLO	61.0	0	0	0	NaN	0	NaN	29	False
3	2	10006	1874946579	R	CC	61.0	0	0	0	NaN	0	NaN	29	False
4	2	10011	220375232	L	CC	55.0	0	0	0	0.0	0	NaN	21	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
54701	1	9973	1729524723	R	MLO	43.0	0	0	0	1.0	0	C	49	False
54702	1	9989	63473691	L	MLO	60.0	0	0	0	NaN	0	C	216	False
54703	1	9989	1078943060	L	CC	60.0	0	0	0	NaN	0	C	216	False
54704	1	9989	398038886	R	MLO	60.0	0	0	0	0.0	0	C	216	True
54705	1	9989	439796429	R	CC	60.0	0	0	0	0.0	0	C	216	True

54706 rows × 14 columns

```python
import pandas as pd

# Merge the dataframes on the "file_name" column
merged_df = pd.merge(df0, df1, on='file_name')

# Check the merged DataFrame
print(merged_df.shape)
merged_df
```

```
(54706, 33)
```

	file_name	site_id	patient_id	image_id	laterality	view	age	cancer	biopsy	invasive	...	dissimilarity
0	10006_462822612.png	2	10006	462822612	L	CC	61.0	0	0	0	...	2.006679
1	10006_1459541791.png	2	10006	1459541791	L	MLO	61.0	0	0	0	...	2.785983
2	10006_1864590858.png	2	10006	1864590858	R	MLO	61.0	0	0	0	...	2.556648
3	10006_1874946579.png	2	10006	1874946579	R	CC	61.0	0	0	0	...	1.844593
4	10011_220375232.png	2	10011	220375232	L	CC	55.0	0	0	0	...	3.131832
...	...	...	...	...	...	...	...	...	...	...	...	...
54701	9973_1729524723.png	1	9973	1729524723	R	MLO	43.0	0	0	0	...	10.118842
54702	9989_63473691.png	1	9989	63473691	L	MLO	60.0	0	0	0	...	1.556541
54703	9989_1078943060.png	1	9989	1078943060	L	CC	60.0	0	0	0	...	0.943275
54704	9989_398038886.png	1	9989	398038886	R	MLO	60.0	0	0	0	...	1.577145
54705	9989_439796429.png	1	9989	439796429	R	CC	60.0	0	0	0	...	0.933778

54706 rows × 33 columns

```
# # Drop the first 4 columns
# merged_df = merged_df.drop(merged_df.columns[:4], axis=1)

# # Drop the 'machine_id' column
merged_df = merged_df.drop('file_name', axis=1)
merged_df = merged_df.drop('patient_id', axis=1)
merged_df = merged_df.drop('image_id', axis=1)

# # Display the updated DataFrame
merged_df
```

	site_id	laterality	view	age	cancer	biopsy	invasive	BIRADS	implant	density	...	dissim
0	2	L	CC	61.0	0	0	0	NaN	0	NaN	...	
1	2	L	MLO	61.0	0	0	0	NaN	0	NaN	...	
2	2	R	MLO	61.0	0	0	0	NaN	0	NaN	...	
3	2	R	CC	61.0	0	0	0	NaN	0	NaN	...	
4	2	L	CC	55.0	0	0	0	0.0	0	NaN	...	
...	...	...	...	...	...	...	...	...	...	...	...	
54701	1	R	MLO	43.0	0	0	0	1.0	0	C	...	1
54702	1	L	MLO	60.0	0	0	0	NaN	0	C	...	
54703	1	L	CC	60.0	0	0	0	NaN	0	C	...	
54704	1	R	MLO	60.0	0	0	0	0.0	0	C	...	
54705	1	R	CC	60.0	0	0	0	0.0	0	C	...	

54706 rows × 30 columns

Shakshi

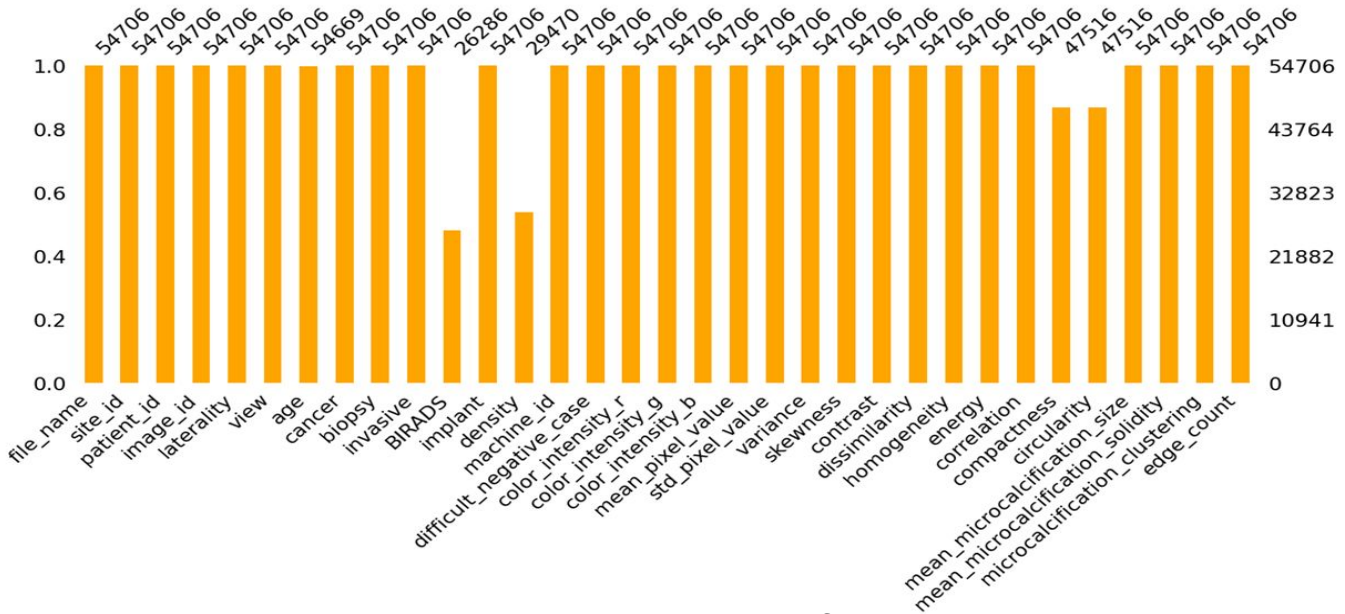
Exploratory Data Analysis

•EDA stands for exploratory data analysis, which is the process of analyzing and summarizing data sets to extract useful information and insights. While performing EDA, We came across several relationships between data but 2 of them were most promising.

1.There is a strong relationship between age and cancer. As people age, their risk of developing cancer increases. According to American cancer society[1] breast cancer mainly occurs in middle-aged and older women. The median age at the time of breast cancer diagnosis is 62. This finding is consistent with our findings in this dataset.
2.A biopsy is a medical procedure that involves the removal of a small sample of tissue or cells from the body for examination under a microscope. Therefore, cancer and biopsy both have strong correlation with each other.

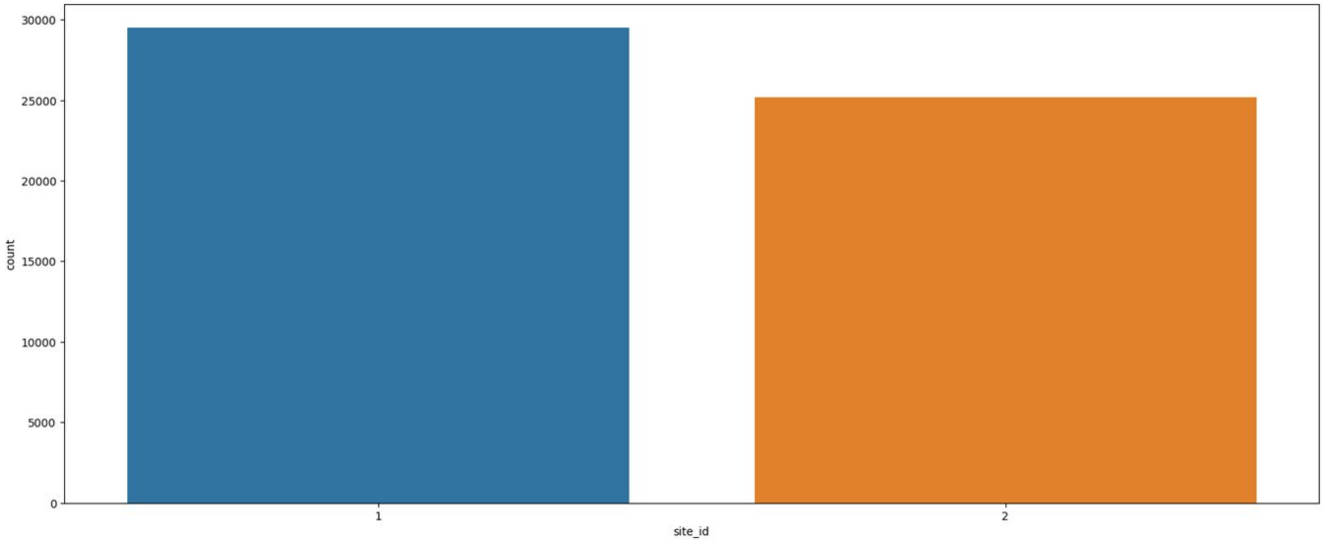
EDA

Missing values



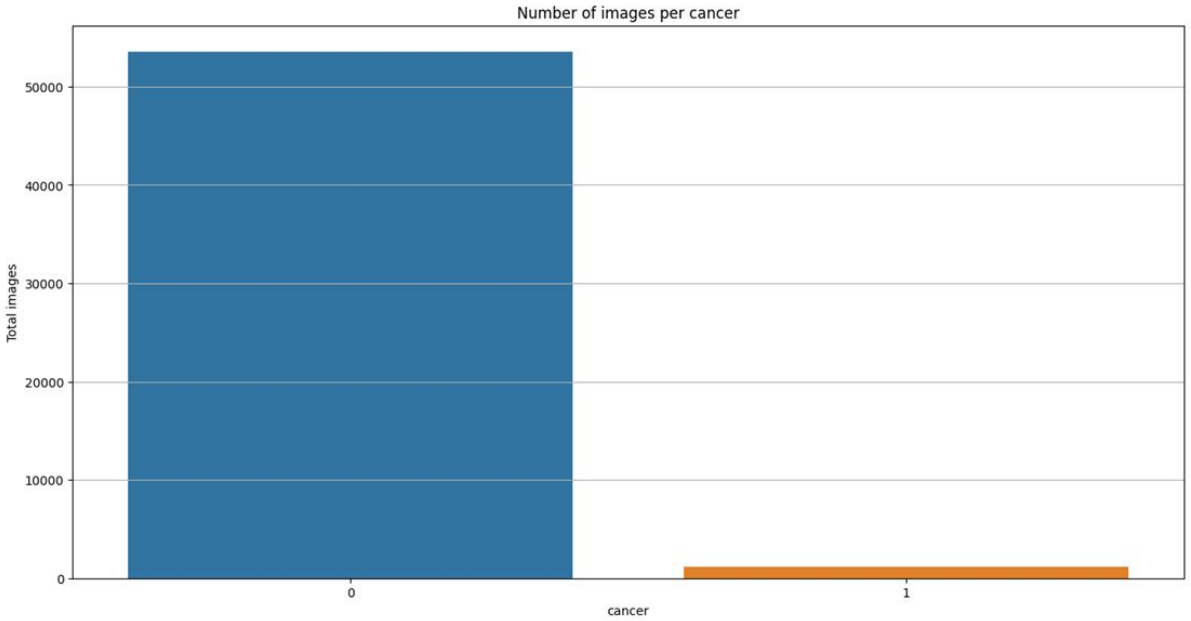
There are multiple missing values in our dataset. All of these missing values are limited to 4 columns. It is really important to find missing values in our dataset because missing data can introduce bias and reduce the accuracy of our analysis.

Site_id



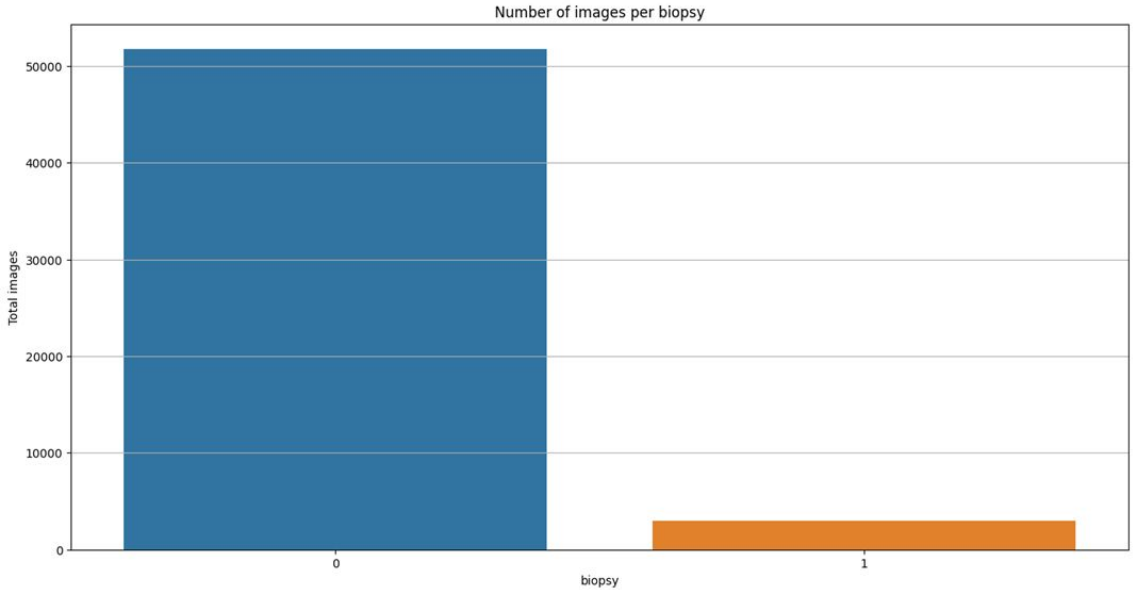
The `site_id` attribute is important because it provides information about the source hospital that collected the data. This information can be useful in determining if there are any variations in the data collection process across different hospitals, which may affect the quality and reliability of the data. For example, certain hospitals may have different imaging devices, screening protocols, or biopsy practices that could potentially impact the accuracy of the breast cancer diagnosis. Understanding the source hospital can help researchers identify and control for any such differences, and potentially improve the quality of their analyses and predictions. Additionally, `site_id` can be useful for tracking the origin of the data, and for ensuring the privacy and security of patient information.

Number of cases for cancer



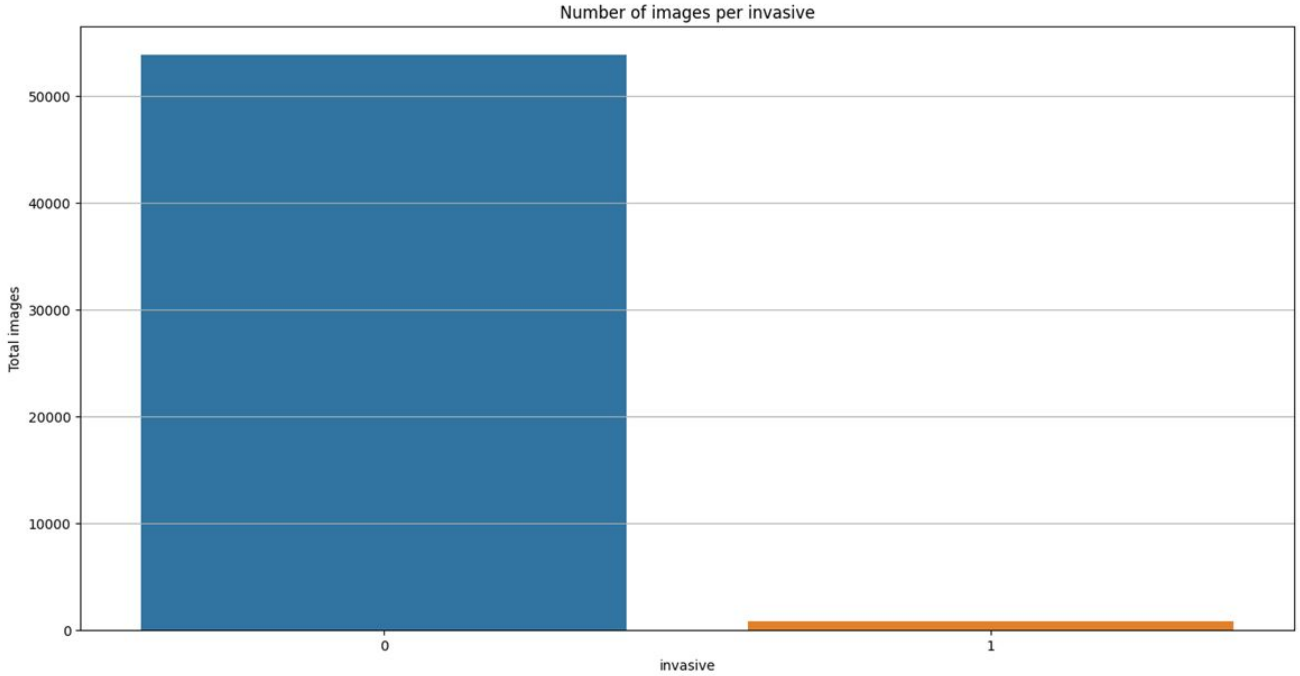
It was seen that most of the images correspond to Mammographies of non-cancer patients. This could be a challenge for the overall prediction as the percentage of cancer images is low.

Biopsy



Biopsy also showed a similar trend where all the patients who underwent biopsy did not necessarily have cancer.

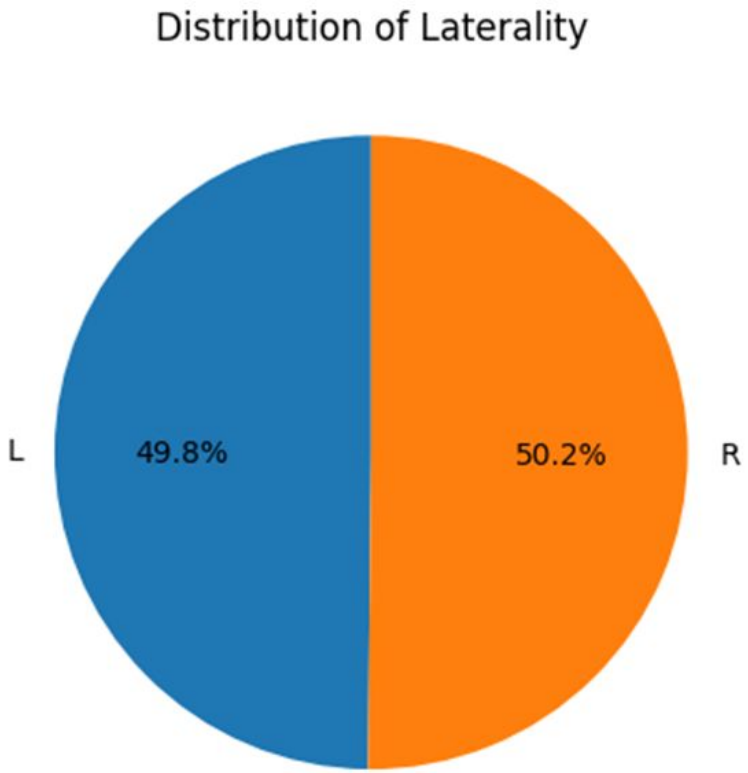
Number of cancer for invasiveness



It can be observed that not all the cancer images represent an invasive/harmful disease.

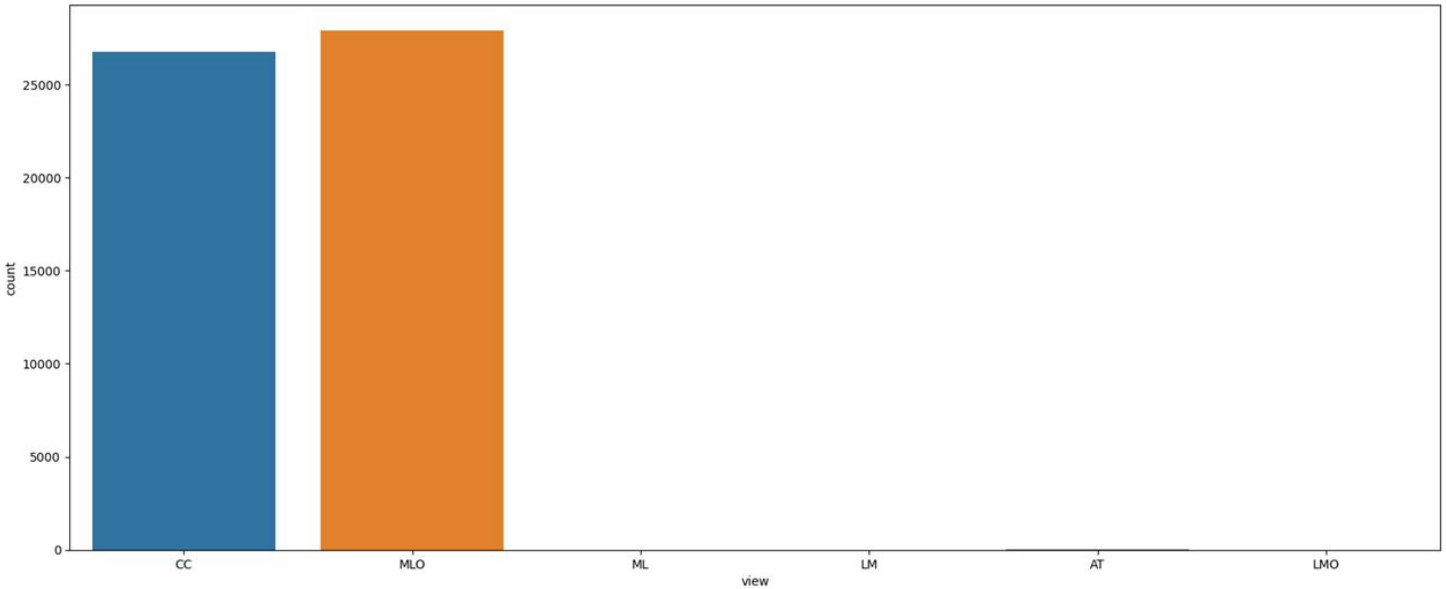
Meghna

Distribution of laterality



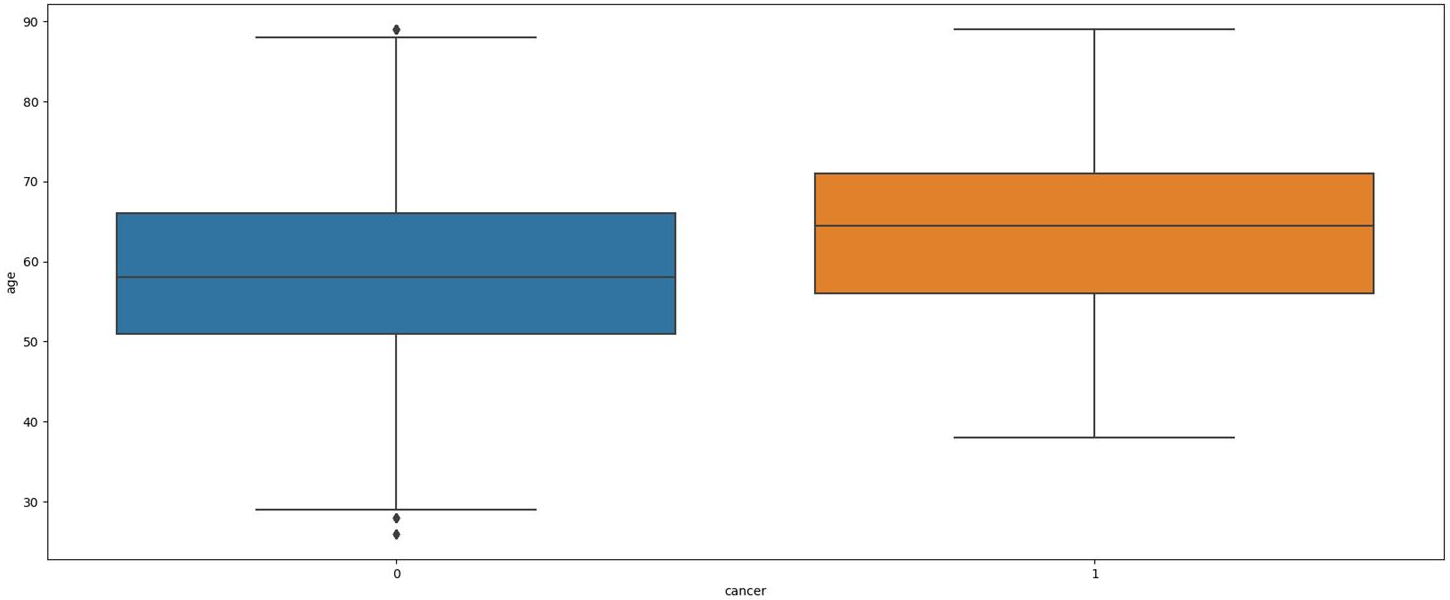
The number of images for each laterality is almost equal for all the patients. This shows that screening Mammography is done on both breasts. This is a positive sign as it reduces the potential for biases or underrepresentation of either side. It also means that any differences or asymmetries in the prevalence or severity of breast cancer between the left and right sides can be more accurately assessed and analyzed.

Images per view



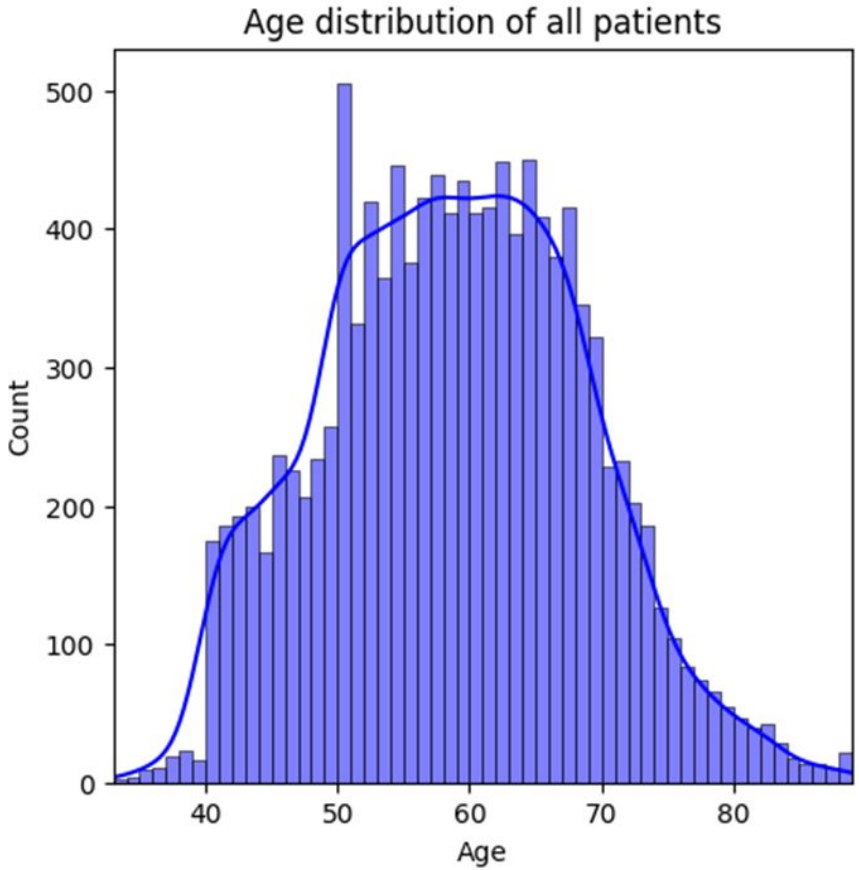
Majority of data is taken in CC (cranial-caudal) or MLO (mediolateral oblique) view. ***This suggests that these two views are commonly used in breast imaging and are likely important for accurate diagnosis and assessment of breast health.*** The extremely low count of images taken in the AT (axial) view suggests that this view may not be commonly used in breast imaging or may not be as useful for assessing breast health as the CC and MLO views.

Outliers for age



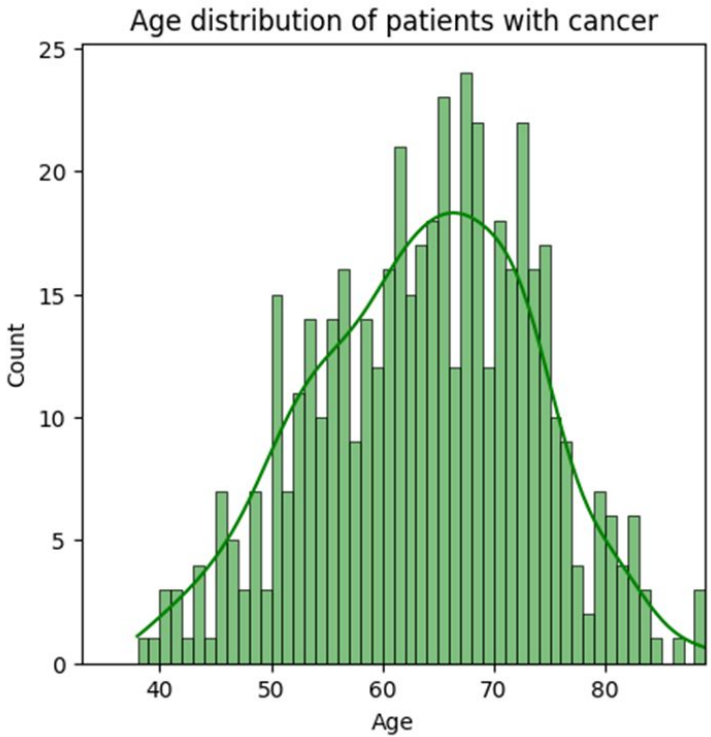
We notice few outliers specifically the patients that are below the age of 30 and near 90 years of age.

Age distribution of all patients



We notice a normal distribution. A normal distribution of age for all patients suggests that the ages of the patients in the dataset are evenly distributed around a central value, with relatively few patients at the extremes of the age range.

Age distribution of all patients with cancer



We can observe that the age of patients with cancer is negatively skewed. This means that most of the patients who had cancer were near the age of 60.

```
Mean age (cancer) is: 63.49382716049383
Mode of age (cancer) is: 0    67.0
Name: age, dtype: float64
Mean age (non-cancer) is: 58.63854105387007
Mode of age (non-cancer) is: 0    50.0
Name: age, dtype: float64
```

According to the most recent statistical data from National Cancer Institute Surveillance, Epidemiology, and End Results (SEER) Program, the median age of a cancer diagnosis is 66 years. This means that half of cancer cases occur in people below this age and half in people above this age. A similar pattern is seen for many common cancer types. For example, the median age at diagnosis is 62 years for breast cancer, 67 years for colorectal cancer, 71 years for lung cancer, and 66 years for prostate cancer.

•The mean and mode age of patients with cancer and without cancer in the dataset can provide insights into the typical age range of patients with and without cancer.

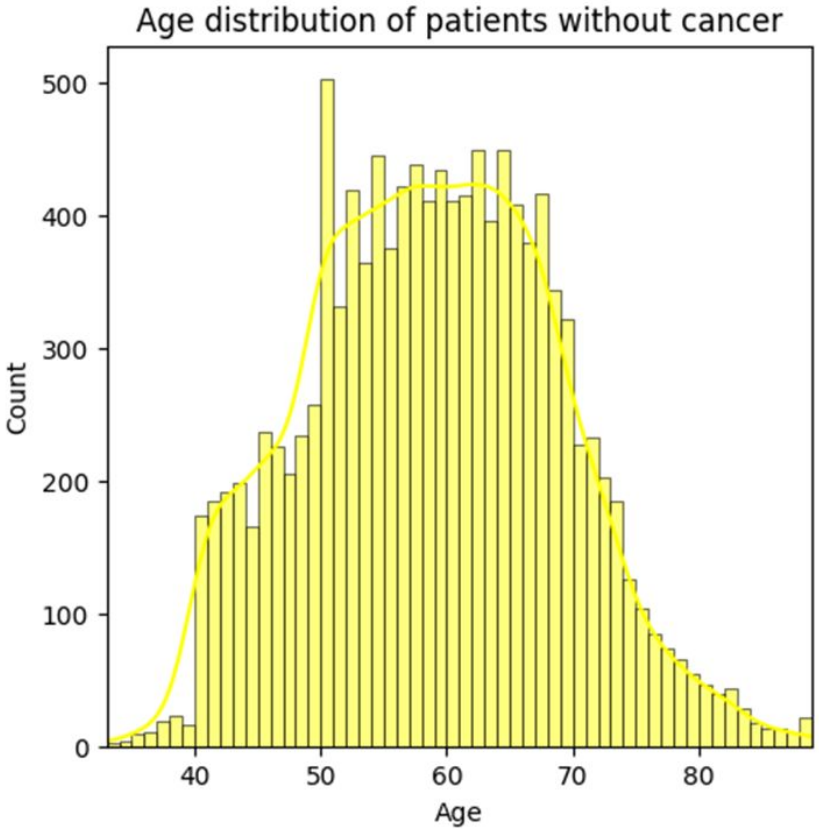
•The mean age of patients with cancer is 63.49 years, while the mean age of patients without cancer is 58.64 years. This suggests that patients with cancer tend to be older than those without cancer.

•The mode age of patients with cancer is 67 years, while the mode age of patients without cancer is 50 years. The mode represents the most common value in the dataset, so this indicates that the most common age for patients with cancer is 67 years, while the most common age for patients without cancer is 50 years.

•These findings are in line with what is known about breast cancer incidence, which increases with age, with the majority of cases diagnosed in women over 50 years old. Therefore, it is important to take age into account when analyzing breast cancer data and developing machine learning models to ensure that the models accurately reflect the age distribution of the real-world population.

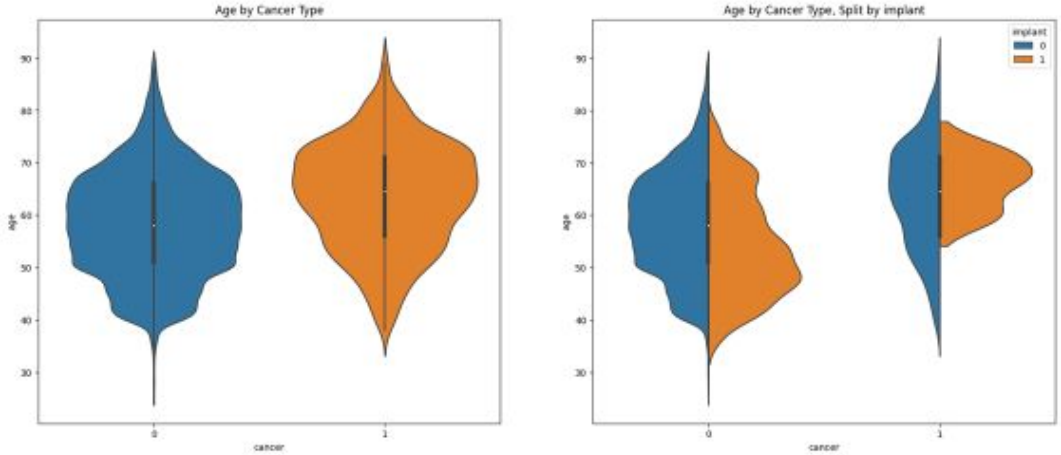
•**So we can conclude that age is an important factor for cancer.**

Age distribution of all patients without cancer



Age distribution of patients without cancer follows a normal distribution

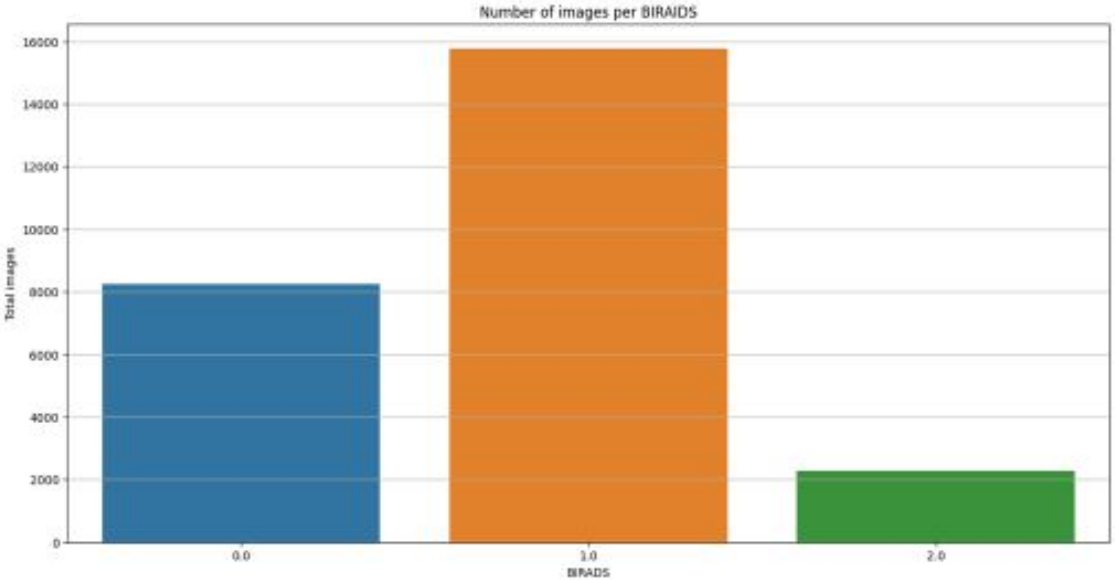
Age by cancer type, split by implant



1.In the violin plot for age by cancer type, we can conclude that most of the females who had cancer were distributed around the age of 70.

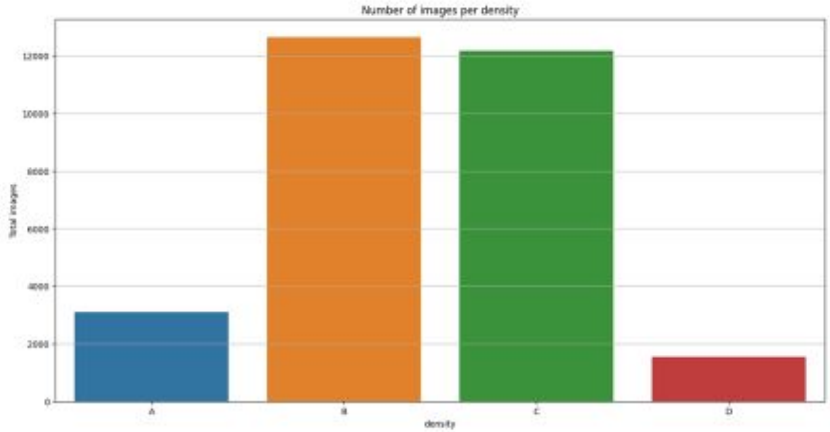
2.In the violin plot for age by cancer type ; split by implant, we can conclude that the patients who had implants might be more prone to cancer as the width of the violen plot for patients with cancer and implants had higher density of observations. Most of these patients were also around the age of 65.

Number of images per BIRAIDS



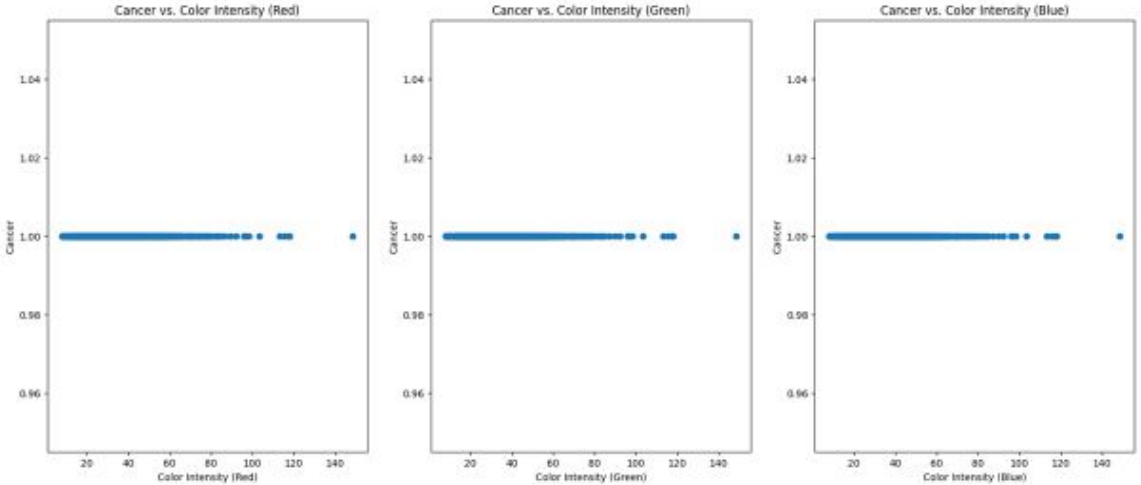
We can observe that the majority of the individuals in the dataset (60%) have a BIRADS score of 1, which indicates a very low suspicion for malignancy. This suggests that the dataset may be skewed towards individuals with a low risk of breast cancer. Usually a BIRADS score of 4 is considered to be abnormal.

Number of images per density



The density variable shows that most of the patients have a tissue density in the categories B and C while the number with D is the lowest.

Cancer vs color intensity



Initially, we did not realise that the mammograms images are black and white. During EDA, we realised that the color intensity for red, green, blue will be the same for black and white images so these fields have no role to play.

Cancer vs contrast

```
Mean contrast for cancer patients 185.11989355497647
Mean contrast for non-cancer patients 162.88073415549715
t-statistic: 7.683347737356626
p-value: 3.196237552382825e-14
```

Null hypothesis - No significant difference in the mean contrast values between patients with and without cancer
Alternate hypothesis - Significant difference in the mean contrast values between patients with and without cancer
Based on the t-statistic and p-value, we can reject the null hypothesis and conclude that there is a statistically significant difference in the mean contrast values between patients with and without cancer. Furthermore, since the t-statistic is positive, we can conclude that the average contrast values for patients with cancer is higher than for patients without cancer.

Cancer vs energy

```
Mean energy for cancer patients 0.6245554721976498
Mean energy for non-cancer patients 0.6241776168818292
t-statistic: 0.22193668943792427
p-value: 0.8244004497061043
```

The t-statistic of 0.2219 and p-value of 0.8244 suggest that there is no statistically significant difference in mean energy values between cancer and non-cancer patients. Therefore, we cannot conclude that there is a relationship between cancer and mean energy values.

Cancer vs compactness

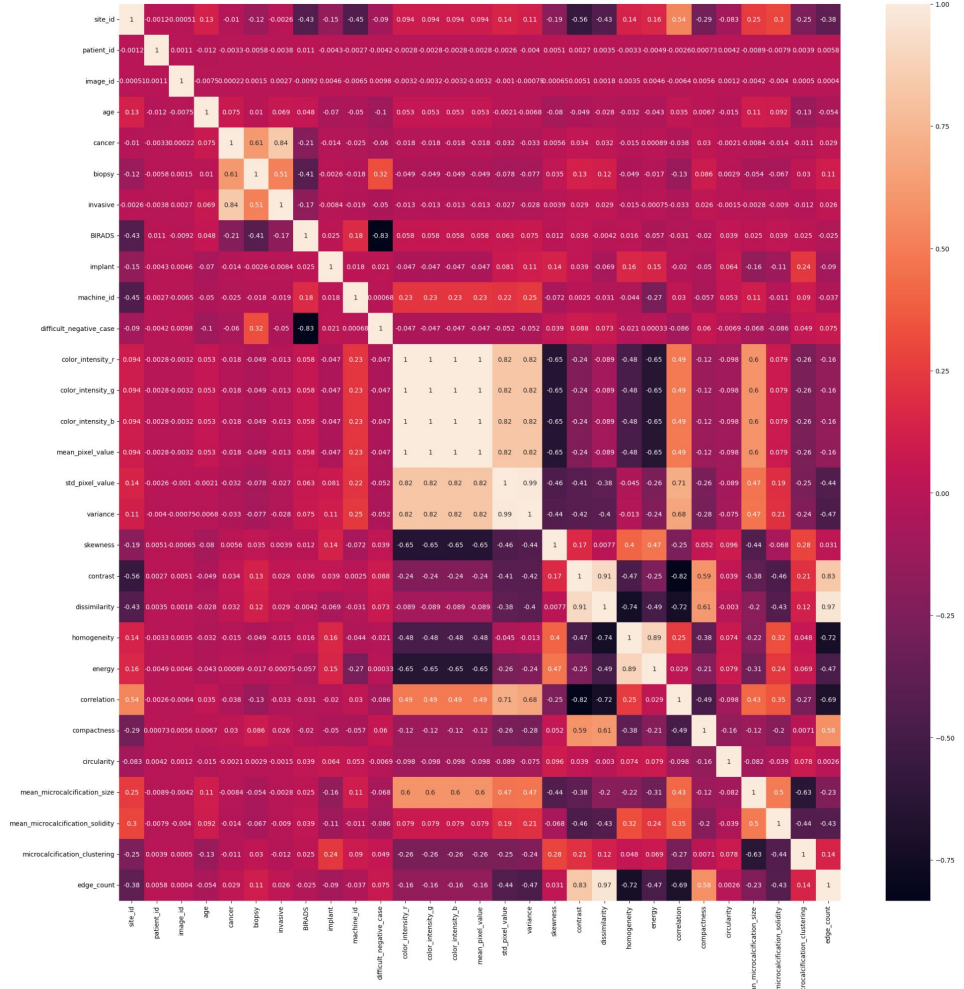
```
Mean compactness for cancer patients: 43.40574120610515
Mean compactness for non - cancer patients: 32.47957550676845
```

Compactness is a measure of how closely pixels are grouped together in an image, and breast tumors can be highly compact as they often appear as dense masses. Therefore, high compactness values may be indicative of the presence of a tumor in mammogram images.

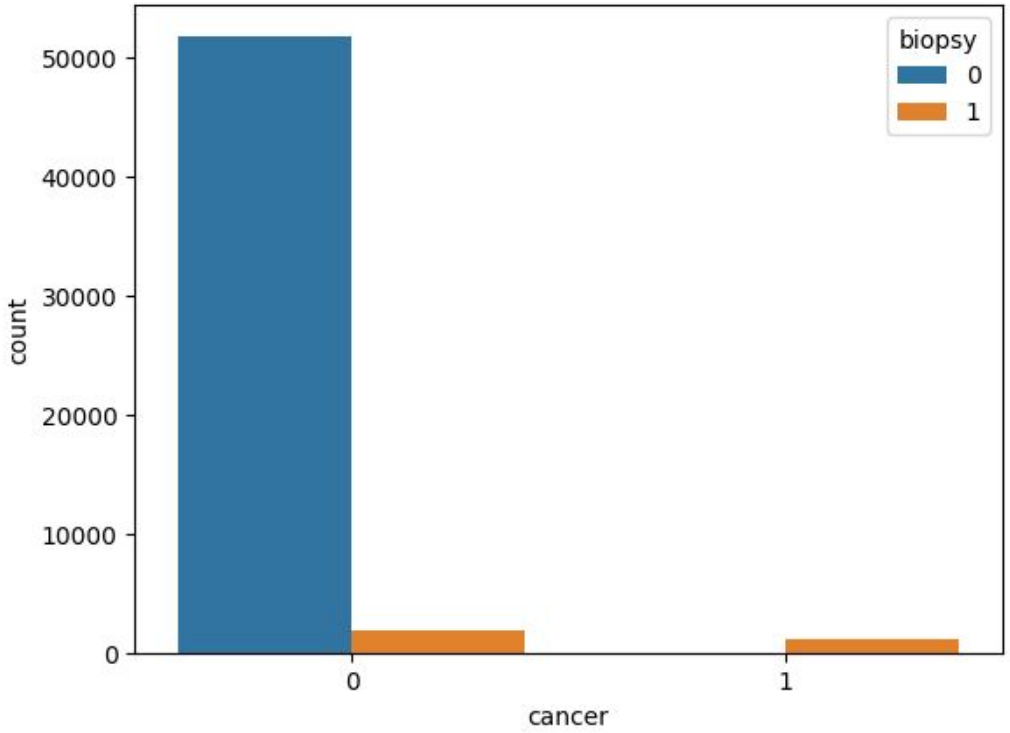
Cancer vs circularity

```
Mean circularity for cancer patients: 1.1849735411273787
Mean circularity for non - cancer patients: 1.2557135854872612
```

Circularity is a shape descriptor that quantifies how close an object's shape is to a perfect circle. In mammograms, circularity can be used to analyze the shape of masses or lesions detected in the breast tissue. A perfectly circular mass would have a circularity value of 1.0, while a value of 0.0 would represent an object with no circularity at all.


site_id
patient_id
image_id
age
cancer
biopsy
invasive
BIRADS
implant
machine_id
difficult_negative_case
color_intensity_r
color_intensity_g
color_intensity_b
mean_pixel_value
std_pixel_value
variance
skewness
contrast
dissimilarity
homogeneity
energy
correlation
compactness
circularity
mean_microcalcification_size
mean_microcalcification_solidity
microcalcification_clustering
edge_count
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Cancer vs biopsy



We can see that biopsy does have a strong correlation to cancer. If cancer is not present, biopsy has also shown the same output for most of the data. And when cancer is presnt, we can see that biopsy has also returned the same value.

Conclusion for EDA

1. There is a strong relationship between age and cancer. As people age, their risk of developing cancer increases. According to American cancer society breast cancer mainly occurs in middle-aged and older women. The median age at the time of breast cancer diagnosis is 62. This finding is consistent with our findings in this dataset.

2. A biopsy is a medical procedure that involves the removal of a small sample of tissue or cells from the body for examination under a microscope. Biopsies are often used to diagnose cancer, as they can provide information about the presence of cancerous cells and their characteristics.
   The relationship between cancer and biopsy is therefore very important. A biopsy is typically recommended when cancer is suspected based on symptoms or imaging tests, such as X-rays, CT scans, or MRIs. By examining the biopsy sample, doctors can confirm the presence of cancer, determine its type and stage, and develop an appropriate treatment plan.

Kapil

Remove – Nan values

```python
# Count NaN values in each column
nan_counts = df.isna().sum()

# Display the NaN counts for each feature
for feature, count in nan_counts.iteritems():
    print(f"NaN count in {feature}: {count}")
```

```
NaN count in site_id: 0
NaN count in laterality: 0
NaN count in view: 0
NaN count in age: 37
NaN count in cancer: 0
NaN count in biopsy: 0
NaN count in invasive: 0
NaN count in BIRADS: 28420
NaN count in implant: 0
NaN count in density: 25236
NaN count in machine_id: 0
NaN count in difficult_negative_case: 0
NaN count in color_intensity_r: 0
NaN count in color_intensity_g: 0
NaN count in color_intensity_b: 0
NaN count in mean_pixel_value: 0
NaN count in std_pixel_value: 0
NaN count in variance: 0
NaN count in skewness: 0
NaN count in contrast: 0
NaN count in dissimilarity: 0
NaN count in homogeneity: 0
NaN count in energy: 0
NaN count in correlation: 0
NaN count in compactness: 7190
NaN count in circularity: 7190
NaN count in mean_microcalcification_size: 0
NaN count in mean_microcalcification_solidity: 0
NaN count in microcalcification_clustering: 0
NaN count in edge_count: 0
<ipython-input-110-e9dd01f713b7>:5: FutureWarning: iteritems
  for feature, count in nan_counts.iteritems():
```

Remove – Nan values

Handle NAN values

This code uses K-Nearest Neighbors (KNN) imputation for both continuous and categorical columns. Note that KNN imputation is a more computationally expensive method than simple imputations like mean, median, or mode, and its performance may depend on the specific characteristics of the data. Make sure to validate the performance of KNN imputation on your dataset before using it in a production setting.

```
# missing / Nan values BEFORE imputation
print(df.isna().sum())
```

```
site_id                              0
laterality                           0
view                                 0
age                                 37
cancer                               0
biopsy                               0
invasive                             0
BIRADS                           28420
implant                              0
density                          25236
machine_id                           0
difficult_negative_case              0
color_intensity_r                    0
color_intensity_g                    0
color_intensity_b                    0
mean_pixel_value                     0
std_pixel_value                      0
variance                             0
skewness                             0
contrast                             0
dissimilarity                        0
homogeneity                          0
energy                               0
correlation                          0
compactness                       7190
circularity                       7190
mean_microcalcification_size         0
mean_microcalcification_solidity     0
microcalcification_clustering        0
edge_count                           0
dtype: int64
```

```python
import pandas as pd
import numpy as np
from sklearn.impute import KNNImputer

# Function to perform KNN imputation for categorical columns
def knn_impute_categorical(df, column, knn_imputer):
    column_encoded = pd.get_dummies(df[column])
    column_imputed = knn_imputer.fit_transform(column_encoded)
    column_imputed_df = pd.DataFrame(column_imputed, columns=column_encoded.columns)
    return column_imputed_df.idxmax(axis=1)

# Function to perform KNN imputation for continuous columns
def knn_impute_continuous(df, column, knn_imputer):
    column_imputed = knn_imputer.fit_transform(df[column].values.reshape(-1, 1))
    return np.round(column_imputed).reshape(-1)

# Set up KNN imputer
knn_imputer = KNNImputer(n_neighbors=3, weights='distance')

# Iterate through columns and apply KNN imputation
for col in df.columns:
    if col in ['cancer', 'biopsy', 'invasive', 'BIRADS', 'implant']:
        df[col] = knn_impute_categorical(df, col, knn_imputer)
    elif col in ['age', 'color_intensity_r', 'color_intensity_g', 'color_intensity_b', 'mean_pixel_value', 'std_pixel_va
        df[col] = knn_impute_continuous(df, col, knn_imputer).flatten()
    elif col in ['laterality', 'view', 'density', 'difficult_negative_case']:
        df[col] = knn_impute_categorical(df, col, knn_imputer)

# Check if there are any remaining missing values
print(df.isna().sum())
```

```
site_id                          0
laterality                       0
view                             0
age                              0
cancer                           0
biopsy                           0
invasive                         0
BIRADS                           0
implant                          0
density                          0
machine_id                       0
difficult_negative_case          0
color_intensity_r                0
color_intensity_g                0
color_intensity_b                0
mean_pixel_value                 0
std_pixel_value                  0
variance                         0
skewness                         0
contrast                         0
dissimilarity                    0
homogeneity                      0
energy                           0
correlation                      0
compactness                      0
circularity                      0
mean_microcalcification_size     0
mean_microcalcification_solidity 0
microcalcification_clustering    0
edge_count                       0
dtype: int64
```

Pre - Processed Data

	site_id	laterality	view	age	cancer	biopsy	invasive	BIRADS	implant	density	...	dissimilarity	homogeneity	energy	correlation	compactness	circ
0	2	L	CC	61.0	0	0	0	0.0	0	A	...	2.0	1.0	1.0	1.0	27.0	
1	2	L	MLO	61.0	0	0	0	0.0	0	A	...	3.0	1.0	1.0	1.0	3.0	
2	2	R	MLO	61.0	0	0	0	0.0	0	A	...	3.0	1.0	1.0	1.0	29.0	
3	2	R	CC	61.0	0	0	0	0.0	0	A	...	2.0	1.0	1.0	1.0	27.0	
4	2	L	CC	55.0	0	0	0	0.0	0	A	...	3.0	1.0	1.0	1.0	33.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
54701	1	R	MLO	43.0	0	0	0	1.0	0	C	...	10.0	0.0	0.0	1.0	243.0	
54702	1	L	MLO	60.0	0	0	0	0.0	0	C	...	2.0	1.0	1.0	1.0	2.0	
54703	1	L	CC	60.0	0	0	0	0.0	0	C	...	1.0	1.0	0.0	1.0	1.0	
54704	1	R	MLO	60.0	0	0	0	0.0	0	C	...	2.0	1.0	0.0	1.0	2.0	
54705	1	R	CC	60.0	0	0	0	0.0	0	C	...	1.0	1.0	0.0	1.0	1.0	

54706 rows × 30 columns

Pre - Processed Data

```
[33] # Preprocessing
     laterality_encoder = LabelEncoder()
     view_encoder = LabelEncoder()
     density_encoder = LabelEncoder()
     difficult_negative_case_encoder = LabelEncoder()

     df['laterality'] = laterality_encoder.fit_transform(df['laterality'].astype(str))
     df['view'] = view_encoder.fit_transform(df['view'].astype(str))
     df['density'] = density_encoder.fit_transform(df['density'].astype(str))
     df['difficult_negative_case'] = difficult_negative_case_encoder.fit_transform(df['difficult_negative_case'].astype(str))

     # Scaling continuous features
     continuous_features = ['age', 'color_intensity_r', 'color_intensity_g', 'color_intensity_b', 'mean_pixel_value', 'std_pixel_value', 'variance', 'skewness',
     minmax_scaler = MinMaxScaler()
     scaled_data_minmax = minmax_scaler.fit_transform(df[continuous_features])
     scaled_df_minmax = pd.DataFrame(scaled_data_minmax, columns=continuous_features)
     df[continuous_features] = scaled_df_minmax
```

Post- Processed Data

```
df
```

	site_id	laterality	view	age	cancer	biopsy	invasive	BIRADS	implant	density	...	dissimilarity	homogeneity	energy	correlation	compactness
0	2	0	1	0.555556	0	0	0	0.0	0	0	...	0.090909	1.0	1.0	1.0	0.041284
1	2	0	5	0.555556	0	0	0	0.0	0	0	...	0.136364	1.0	1.0	1.0	0.004587
2	2	1	5	0.555556	0	0	0	0.0	0	0	...	0.136364	1.0	1.0	1.0	0.044343
3	2	1	1	0.555556	0	0	0	0.0	0	0	...	0.090909	1.0	1.0	1.0	0.041284
4	2	0	1	0.460317	0	0	0	0.0	0	0	...	0.136364	1.0	1.0	1.0	0.050459
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
54701	1	1	5	0.269841	0	0	0	1.0	0	2	...	0.454545	0.0	0.0	1.0	0.371560
54702	1	0	5	0.539683	0	0	0	0.0	0	2	...	0.090909	1.0	1.0	1.0	0.003058
54703	1	0	1	0.539683	0	0	0	0.0	0	2	...	0.045455	1.0	0.0	1.0	0.001529
54704	1	1	5	0.539683	0	0	0	0.0	0	2	...	0.090909	1.0	0.0	1.0	0.003058
54705	1	1	1	0.539683	0	0	0	0.0	0	2	...	0.045455	1.0	0.0	1.0	0.001529

54706 rows × 30 columns

```python
# Step 1: Train the 5 models
models = [
    ('Logistic Regression', LogisticRegression()),
    ('Decision Tree', DecisionTreeClassifier()),
    ('Random Forest', RandomForestClassifier()),
    ('K-Nearest Neighbour', KNeighborsClassifier()),
    ('Categorical NB', CategoricalNB())
]

model_metrics = []

for name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_pred_proba = model.predict_proba(X_test)[:, 1]

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, y_pred_proba)
    confusion = confusion_matrix(y_test, y_pred)
    cohen_kappa = cohen_kappa_score(y_test, y_pred)
    matthews_corr = matthews_corrcoef(y_test, y_pred)
    log_loss_score = log_loss(y_test, y_pred_proba)

    metrics = {
        'name': name,
        'accuracy': accuracy,
        'precision': precision,
        'recall': recall,
        'f1': f1,
        'roc_auc': roc_auc,
        'confusion': confusion,
        'cohen_kappa': cohen_kappa,
        'matthews_corr': matthews_corr,
        'log_loss': log_loss_score
    }
    model_metrics.append(metrics)

model_metrics
```

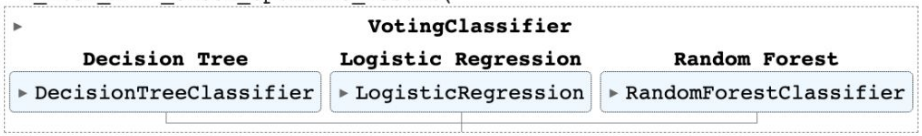
[1] **'accuracy': (TP + TN) / (TP + TN + FP + FN),**
'accuracy': Proportion of correct predictions among total predictions; high accuracy indicates better classifier performance.
[2] **'precision': TP / (TP + FP)**
'precision': Ratio of true positive predictions to total positive predictions; high precision indicates fewer false positives.
'recall': TP / (TP + FN)
'recall': Ratio of true positives to the sum of true positives and false negatives; high recall indicates fewer false negatives.
[3] **'f1': 2 * (precision * recall) / (precision + recall),**
'f1': Harmonic mean of precision and recall; higher F1 score indicates better balance between precision and recall.
[4] **'roc_auc': Area under the receiver operating characteristic (ROC) curve**; higher AUC indicates better classifier performance across various thresholds.
[5] **'confusion': Matrix of [[TP, FP], [FN, TN]],**
'confusion': Matrix representing the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions.
[6] **'cohen_kappa': (observed_accuracy - chance_accuracy) / (1 - chance_accuracy),**
'cohen_kappa': Agreement between predicted and actual labels, adjusted for chance; higher kappa values indicate better agreement.
[7] **'matthews_corr': (TPTN - FPFN) / sqrt((TP+FP)(TP+FN)(TN+FP)*(TN+FN)),**
'matthews_corr': Correlation coefficient between observed and predicted binary classifications; higher values indicate better classifier performance.
[8] **'log_loss': -1 * (y * log(y_pred) + (1 - y) * log(1 - y_pred)) (averaged over samples)**
'log_loss': Measure of the difference between predicted probabilities and true labels; lower values indicate better classifier performance.

0s

```
# Step 2: Find top 3 models
# You can assign weights to each performance metric and compute the weighted average score for each model
# For illustration, we assign equal weights to all metrics
weights = {
    'accuracy': 1,
    'precision': 1,
    'recall': 1,
    'f1': 1,
    'roc_auc': 1,
    'cohen_kappa': 1,
    'matthews_corr': 1,
    'log_loss': -1  # negative weight because lower log_loss is better
}

for metrics in model_metrics:
    weighted_score = 0
    for key, weight in weights.items():
        if key != 'confusion':
            weighted_score += metrics[key] * weight
    metrics['weighted_score'] = weighted_score

model_metrics.sort(key=lambda x: x['weighted_score'], reverse=True)
top3_models = model_metrics[:3]
top3_models
```

```python
from sklearn.ensemble import VotingClassifier
# Step 3: Retrain top 3 models and ensemble model
# Create a dictionary for easy model lookup
model_dict = {name: model for name, model in models}

# Select the top 3 models from the dictionary
top3_model_list = [(model_metrics['name'], model_dict[model_metrics['name']]) for model_metrics in top3_models]

# Create the ensemble model
ensemble_model = VotingClassifier(estimators=top3_model_list, voting='soft')

# Add the ensemble model to the list of models
models.append(('Ensemble', ensemble_model))

# Train the ensemble model
ensemble_model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```



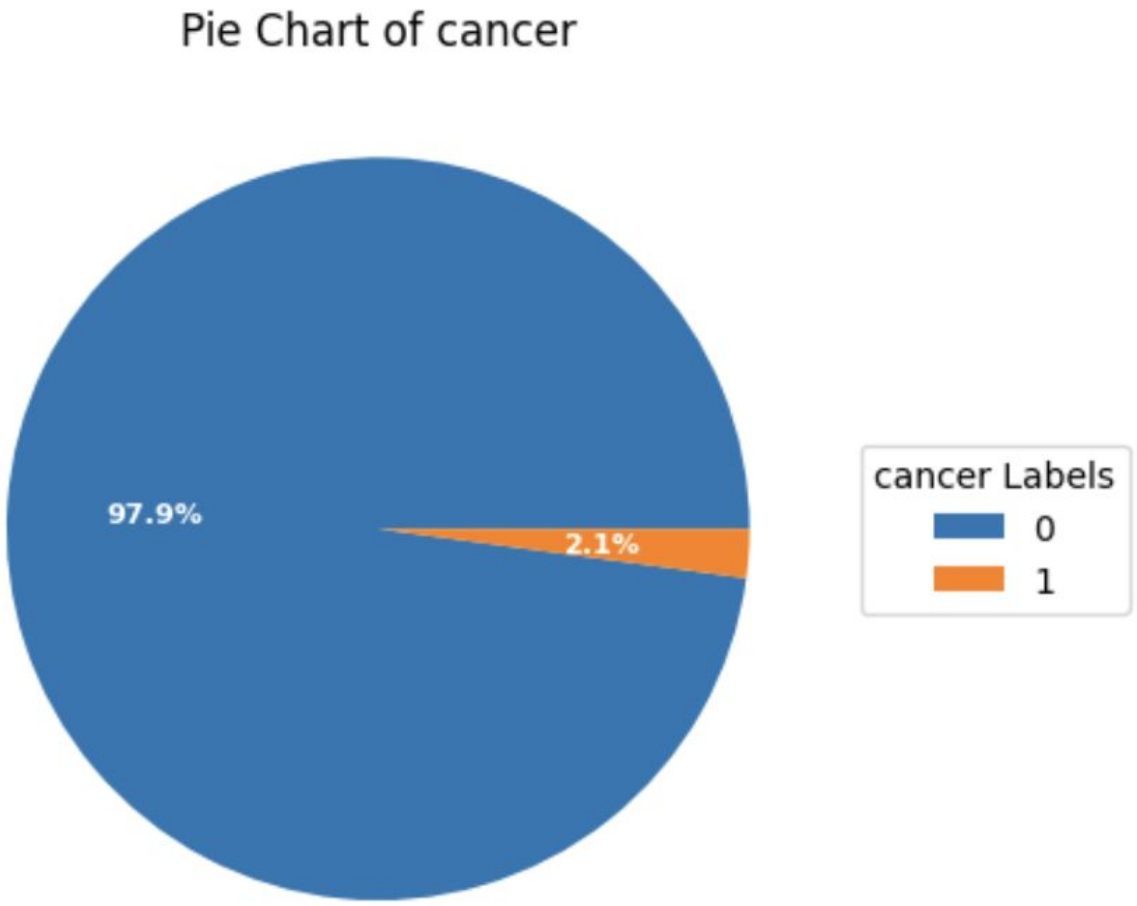
```python
# Step 4: Take a weighted average of each performance metric from the top 3 models and the ensemble model
model_metrics = []

for name, model in models:
    if name in [model_metrics['name'] for model_metrics in top3_models] or name == 'Ensemble':
        y_pred = model.predict(X_test)
        y_pred_proba = model.predict_proba(X_test)[:, 1]

        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred)
        recall = recall_score(y_test, y_pred)
        f1 = f1_score(y_test, y_pred)
        roc_auc = roc_auc_score(y_test, y_pred_proba)
        confusion = confusion_matrix(y_test, y_pred)
        cohen_kappa = cohen_kappa_score(y_test, y_pred)
        matthews_corr = matthews_corrcoef(y_test, y_pred)
        log_loss_score = log_loss(y_test, y_pred_proba)

        metrics = {
            'name': name,
            'accuracy': accuracy,
            'precision': precision,
            'recall': recall,
            'f1': f1,
            'roc_auc': roc_auc,
            'confusion': confusion,
            'cohen_kappa': cohen_kappa,
            'matthews_corr': matthews_corr,
            'log_loss': log_loss_score
        }
        model_metrics.append(metrics)


model_metrics
```

```
[25] # Convert the model_metrics list to a DataFrame
     model_metrics_df = pd.DataFrame(model_metrics)

     # Display the model_metrics DataFrame
     model_metrics_df
```

	name	accuracy	precision	recall	f1	roc_auc	confusion	cohen_kappa	matthews_corr	log_loss
0	Logistic Regression	1.0	1.0	1.0	1.0	1.0	[[10711, 0], [0, 231]]	1.0	1.0	1.037091e-03
1	Decision Tree	1.0	1.0	1.0	1.0	1.0	[[10711, 0], [0, 231]]	1.0	1.0	2.220446e-16
2	Random Forest	1.0	1.0	1.0	1.0	1.0	[[10711, 0], [0, 231]]	1.0	1.0	2.616796e-03
3	Ensemble	1.0	1.0	1.0	1.0	1.0	[[10711, 0], [0, 231]]	1.0	1.0	1.290102e-03

```
[25]
```

Shakshi


Pie Chart of cancer
97.9%
2.1%
cancer Labels
0
1

```python
# Verify the distribution
cancer_percentage = (df0['cancer'] == 1).sum() / len(df0) * 100
non_cancer_percentage = (df0['cancer'] == 0).sum() / len(df0) * 100

print(f"New distribution:")
print(f"df['cancer'] == 0: {non_cancer_percentage:.1f}%")
print(f"df['cancer'] == 1: {cancer_percentage:.1f}%")
print(df0.shape)
```

```
New distribution:
df['cancer'] == 0: 97.9%
df['cancer'] == 1: 2.1%
```

```python
import pandas as pd
import numpy as np

# Load the dataset
df = pd.read_csv('train.csv')

# Find the indices of rows with df['cancer'] == 0
non_cancer_indices = df[df['cancer'] == 0].index

# Calculate 10% of these indices
num_to_change = int(len(non_cancer_indices) * 0.1)

# Randomly select the calculated number of indices
random_indices = np.random.choice(non_cancer_indices, num_to_change, replace=False)

# Change the 'cancer' value to 1 for the selected indices
df.loc[random_indices, 'cancer'] = 1

# Verify the distribution
cancer_percentage = (df['cancer'] == 1).sum() / len(df) * 100
non_cancer_percentage = (df['cancer'] == 0).sum() / len(df) * 100

print(f"New distribution:")
print(f"df['cancer'] == 0: {non_cancer_percentage:.1f}%")
print(f"df['cancer'] == 1: {cancer_percentage:.1f}%")
print(df.shape)
```

```
New distribution:
df['cancer'] == 0: 88.1%
df['cancer'] == 1: 11.9%
```

```
[ ] # Convert the model_metrics list to a DataFrame
    model_metrics_df = pd.DataFrame(model_metrics)

    # Display the model_metrics DataFrame
    model_metrics_df
```

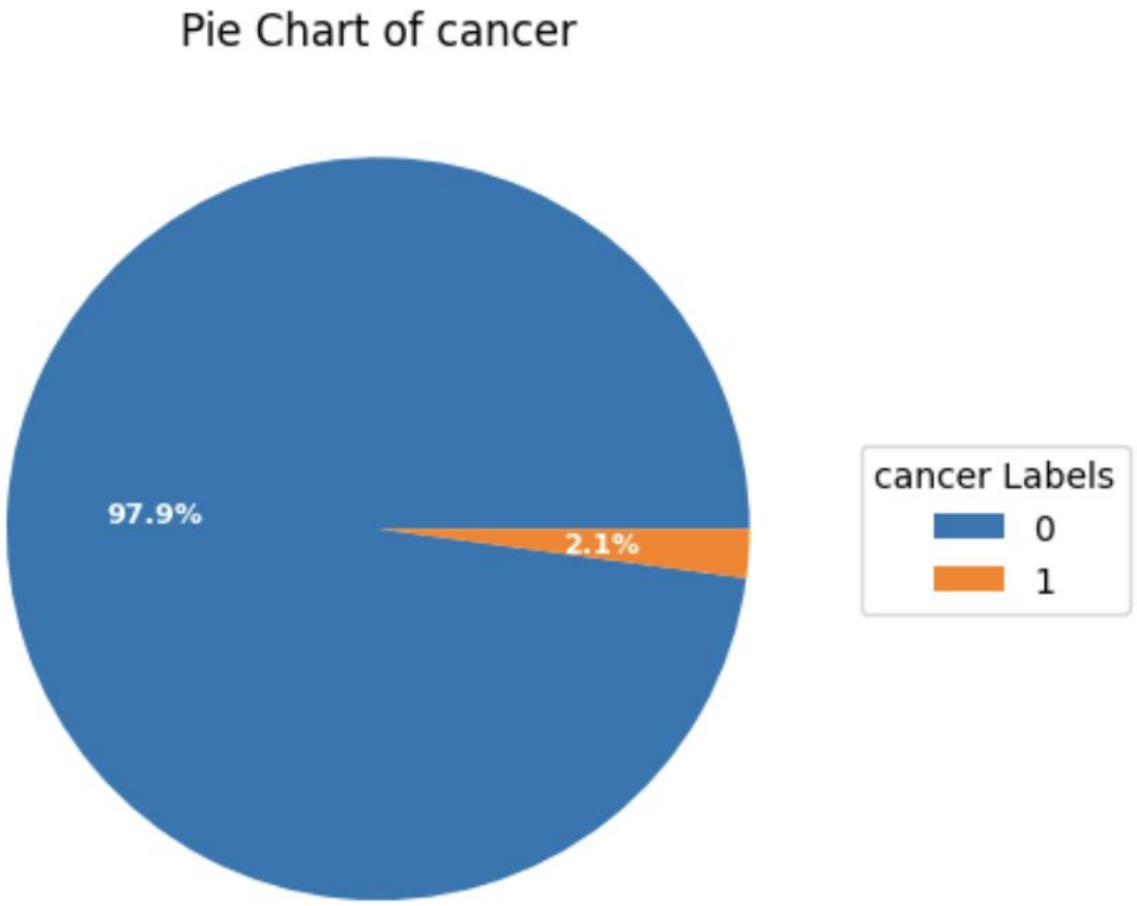
	name	accuracy	precision	recall	f1	roc_auc	confusion	cohen_kappa	matthews_corr	log_loss
0	Logistic Regression	0.896271	1.000000	0.120155	0.214533	0.586115	[[9652, 0], [1135, 155]]	0.194151	0.327891	0.326640
1	Random Forest	0.902851	0.978903	0.179845	0.303864	0.583946	[[9647, 5], [1058, 232]]	0.277420	0.397266	0.347373
2	Categorical NB	0.896363	1.000000	0.120930	0.215768	0.586843	[[9652, 0], [1134, 156]]	0.195298	0.328962	0.329939
3	Ensemble	0.903217	1.000000	0.179070	0.303748	0.593552	[[9652, 0], [1059, 231]]	0.277889	0.401703	0.324627

Vinit

```
[25] # Convert the model_metrics list to a DataFrame
     model_metrics_df = pd.DataFrame(model_metrics)

     # Display the model_metrics DataFrame
     model_metrics_df
```

	name	accuracy	precision	recall	f1	roc_auc	confusion	cohen_kappa	matthews_corr	log_loss
0	Logistic Regression	1.0	1.0	1.0	1.0	1.0	[[10711, 0], [0, 231]]	1.0	1.0	1.037091e-03
1	Decision Tree	1.0	1.0	1.0	1.0	1.0	[[10711, 0], [0, 231]]	1.0	1.0	2.220446e-16
2	Random Forest	1.0	1.0	1.0	1.0	1.0	[[10711, 0], [0, 231]]	1.0	1.0	2.616796e-03
3	Ensemble	1.0	1.0	1.0	1.0	1.0	[[10711, 0], [0, 231]]	1.0	1.0	1.290102e-03

```
[25]
```


Pie Chart of cancer
97.9%
2.1%
cancer Labels
0
1

- Because of highly imbalanced datasets, everything is looking PERFECT

  So, let's train in batches to maintain equal ratio of Cancer and Non-Cancer datasets

Step 1: To create equal batches of cancer and non-cancer cases,

0s

```
import numpy as np

# Create separate datasets for cancer and non-cancer cases
cancer_df = df[df['cancer'] == 1]
non_cancer_df = df[df['cancer'] == 0]

# Calculate the number of batches needed
num_batches = int(non_cancer_df.shape[0] / cancer_df.shape[0])

# Create a list to store balanced datasets
balanced_datasets = []

# Create balanced datasets by sampling from non-cancer cases
for _ in range(num_batches):
    non_cancer_sample = non_cancer_df.sample(n=cancer_df.shape[0], random_state=42)
    balanced_df = pd.concat([cancer_df, non_cancer_sample], axis=0)
    balanced_datasets.append(balanced_df)

#balanced_datasets
```

	batch	name	accuracy	precision	recall	f1	roc_auc	confusion	cohen_kappa	matthews_corr	log_loss
0	0	Logistic Regression	1.000000	1.000000	1.00000	1.000000	1.000000	[[221, 0], [0, 243]]	1.000000	1.000000	1.404645e-02
1	0	Decision Tree	1.000000	1.000000	1.00000	1.000000	1.000000	[[221, 0], [0, 243]]	1.000000	1.000000	2.220446e-16
2	0	Random Forest	1.000000	1.000000	1.00000	1.000000	1.000000	[[221, 0], [0, 243]]	1.000000	1.000000	3.062240e-02
3	0	K-Nearest Neighbour	0.987069	0.983673	0.99177	0.987705	0.999404	[[217, 4], [2, 241]]	0.974069	0.974105	3.230006e-02
4	0	Categorical NB	1.000000	1.000000	1.00000	1.000000	1.000000	[[221, 0], [0, 243]]	1.000000	1.000000	1.197949e-02
...	...	...	...	...	...	...	...	...	...	...	...
225	45	Logistic Regression	1.000000	1.000000	1.00000	1.000000	1.000000	[[221, 0], [0, 243]]	1.000000	1.000000	1.404645e-02
226	45	Decision Tree	1.000000	1.000000	1.00000	1.000000	1.000000	[[221, 0], [0, 243]]	1.000000	1.000000	2.220446e-16
227	45	Random Forest	1.000000	1.000000	1.00000	1.000000	1.000000	[[221, 0], [0, 243]]	1.000000	1.000000	3.027248e-02
228	45	K-Nearest Neighbour	0.987069	0.983673	0.99177	0.987705	0.999404	[[217, 4], [2, 241]]	0.974069	0.974105	3.230006e-02
229	45	Categorical NB	1.000000	1.000000	1.00000	1.000000	1.000000	[[221, 0], [0, 243]]	1.000000	1.000000	1.197949e-02

230 rows × 11 columns

Step 2: Find top 3 models

First, let's modify the code to calculate the average performance for each model across all batches. Then, we can find the top 3 models based on the weighted average score:

[29] 0s

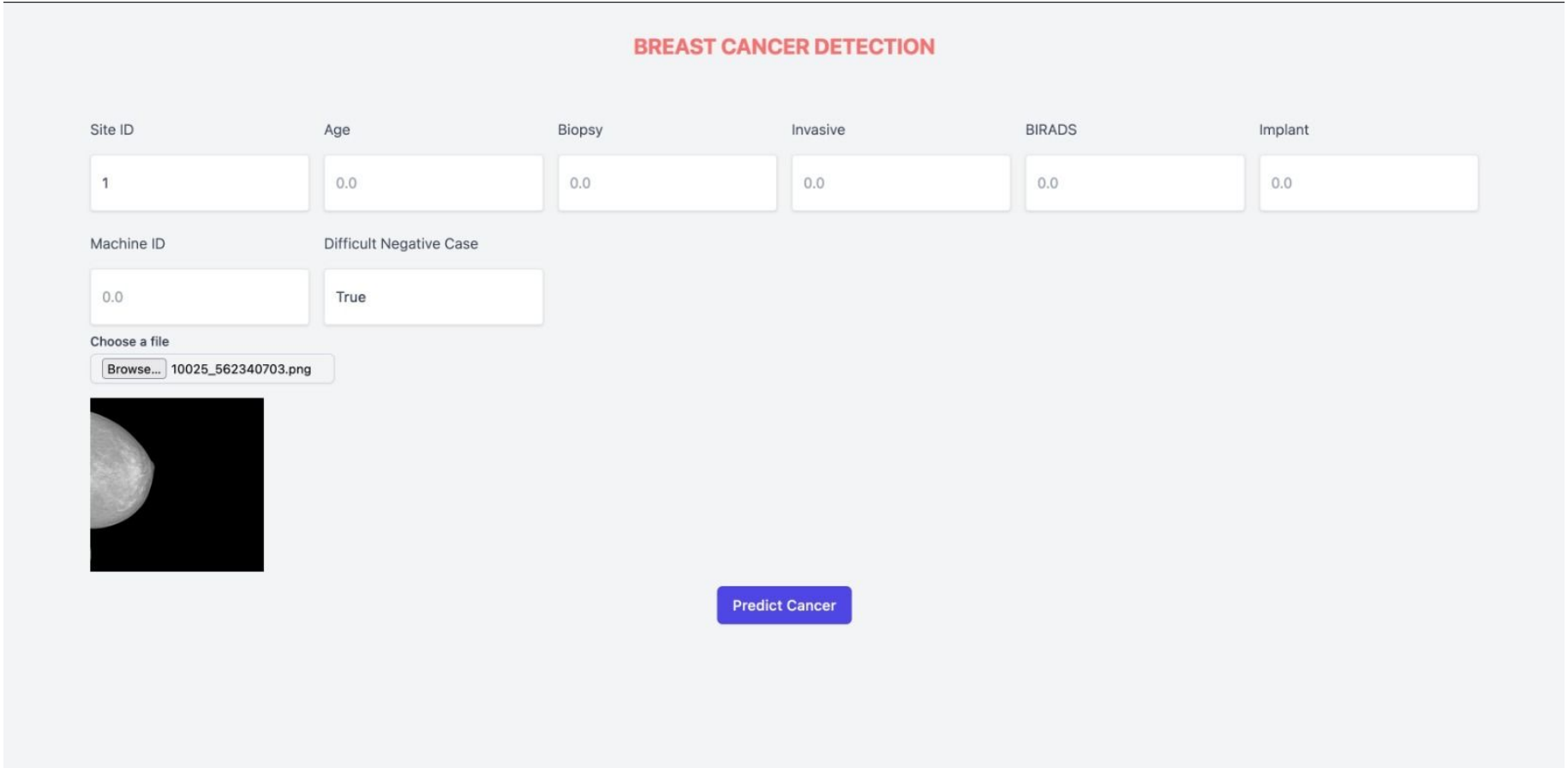
```
# Calculate the average performance for each model across all batches
model_avg_metrics = {}
for metrics in model_metrics:
    name = metrics['name']
    if name not in model_avg_metrics:
        model_avg_metrics[name] = {key: 0 for key in weights.keys()}
        model_avg_metrics[name]['count'] = 0

    for key in weights.keys():
        model_avg_metrics[name][key] += metrics[key]
    model_avg_metrics[name]['count'] += 1

for name, metrics in model_avg_metrics.items():
    for key in weights.keys():
        metrics[key] /= metrics['count']

# Compute the weighted average score for each model
for name, metrics in model_avg_metrics.items():
    weighted_score = 0
    for key, weight in weights.items():
        weighted_score += metrics[key] * weight
    metrics['weighted_score'] = weighted_score

# Sort the models by weighted score and select the top 3
sorted_models = sorted(model_avg_metrics.items(), key=lambda x: x[1]['weighted_score'], reverse=True)
top3_models = [model for model, metrics in sorted_models[:3]]
top3_models
```

```
['Decision Tree', 'Categorical NB', 'Logistic Regression']
```

```python
from sklearn.ensemble import VotingClassifier

# Step 3: Retrain top 3 models and create ensemble model
top3_model_instances = [(name, model) for name, model in models if name in top3_models]
ensemble_model = VotingClassifier(estimators=top3_model_instances, voting='soft')

# Retrain top 3 models and the ensemble model
top3_models.append('Ensemble')
all_models = top3_model_instances + [('Ensemble', ensemble_model)]

model_metrics = []
```

	name	accuracy	precision	recall	f1	roc_auc	confusion	cohen_kappa	matthews_corr	log_loss
0	Logistic Regression	1.0	1.0	1.0	1.0	1.0	[[221, 0], [0, 243]]	1.0	1.0	1.404645e-02
1	Decision Tree	1.0	1.0	1.0	1.0	1.0	[[221, 0], [0, 243]]	1.0	1.0	2.220446e-16
2	Categorical NB	1.0	1.0	1.0	1.0	1.0	[[221, 0], [0, 243]]	1.0	1.0	1.197949e-02
3	Ensemble	1.0	1.0	1.0	1.0	1.0	[[221, 0], [0, 243]]	1.0	1.0	8.553647e-03

FrontEnd


BREAST CANCER DETECTION
Site ID
Age
Biopsy
Invasive
BIRADS
Implant
1
0.0
0.0
0.0
0.0
0.0
Machine ID
Difficult Negative Case
0.0
True
Choose a file
Browse...
10025_562340703.png
Predict Cancer

FrontEnd

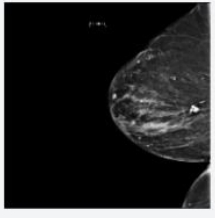
BREAST CANCER DETECTION

Site ID	Laterality	View	Age	Biopsy	Invasive
2	L	MLO	71	1	1

BIRADS	Implant	Density	Machine ID	Difficult Negative Case
0	0	B	49	False

Choose a file

Browse... 10130_388811999.png



Results

Cancer: Benign

Predict Cancer

Shakshi

Creativity : Data Extraction from Images

```
df
```

	file_name	color_intensity_r	color_intensity_g	color_intensity_b	mean_pixel_value	std_pixel_value	variance	skewness	contrast	dissimil
0	24231_1599132094.png	21.669540	21.669540	21.669540	21.669540	41.618134	1732.069065	2.223763	186.391054	4.7
1	10838_591123709.png	80.702255	80.702255	80.702255	80.702255	69.249010	4795.425401	-0.185761	161.732475	3.3
2	27667_830917739.png	45.809540	45.809540	45.809540	45.809540	54.380373	2957.224955	1.319218	279.208655	7.9
3	16339_1765002339.png	21.175934	21.175934	21.175934	21.175934	47.596471	2265.424064	1.864654	31.575797	1.2
4	2606_72547626.png	21.496078	21.496078	21.496078	21.496078	40.860531	1669.583023	2.136658	234.588542	5.2
...	...	...	...	...	...	...	...	...	...	
54701	57444_2114947064.png	30.386978	30.386978	30.386978	30.386978	50.009012	2500.901319	1.742150	449.614308	8.5
54702	1878_189362032.png	35.229370	35.229370	35.229370	35.229370	39.112316	1529.773256	1.029031	325.719899	8.1
54703	45339_1546789692.png	43.123657	43.123657	43.123657	43.123657	74.371804	5531.165159	1.178659	30.847656	1.4
54704	23855_1024724669.png	24.096588	24.096588	24.096588	24.096588	37.214050	1384.885507	1.505538	180.844393	4.9
54705	3525_1032766690.png	71.472580	71.472580	71.472580	71.472580	75.002949	5625.442424	0.199063	64.805438	3.0

54706 rows × 19 columns

Creativity and Innovation:

1) Extracted Features from 55k images
   Highlight the extraction of key features from a dataset of 55,000 mammogram images.
2) Explain the significance of these features in identifying and diagnosing breast cancer.
3) Built an Application

Contribution to society

- Early Detection and Improved Outcomes to help Healthcare Professionals.
- Healthcare Resource Optimization
- Better detection leads to improved treatment outcomes and patient survival rates.
- Public Health and Policy Planning
- Potential to significantly impact society by improving early detection, personalized treatment, and overall patient outcomes.

References

1. https://www.wikipedia.org/
2. https://chat.openai.com/chat
3. https://www.youtube.com/
4. https://www.google.com/

Meghna

Shakshi

Vinit

Kapil