

Predictive Vision Analysis of Breast  
Cancer Cells for Improved Treatment

## Data Science and Analytics

Athens, GA  
2023-2024

## Table of Contents

<b>Introduction</b>	<b>3</b>
<b>Data Dictionary</b>	<b>4</b>
Breast Cancer Risk Factor Dataset	4
Quantitative Breast Cancer Cell Data	5
Images of Breast Cancer Cell Data	5
<b>Purpose</b>	<b>6</b>
<b>Methods</b>	<b>7</b>
<b>Results</b>	<b>8</b>
Exploratory Data Analysis	8
Machine Learning (KNN) Classification Results	13
Computer Vision (ResNet) Cleaning	21
Computer Vision (ResNet) Results	22
<b>Conclusion</b>	<b>29</b>
<b>Next Steps</b>	<b>30</b>
<b>References</b>	<b>31</b>

## Introduction

Breast cancer is the most common cancer occurring in women and the most common cancer overall in the world. There was an estimate of more than 2.26 million new cases of breast cancer in women during 2020 (WCRF International, 2022). Considering the critical importance of prompt treatment, enhancing the speed of diagnosis is imperative for improving patient outcomes. Since breast cancer diagnosis must be both quick and highly accurate, we propose a three-part, data-based solution: a metadata analysis to find high-risk groups based on historical trends, a K-Nearest Neighbors classification to cluster breast cancer factors in cells, and an image-recognition system for classifying cell mammograms and scans as cancerous or non-cancerous. This data analysis—using data retrieved from the National Cancer Institute and a UC Irvine breast cancer study—leverages exploratory data analysis and machine learning techniques to pinpoint the most contributive components of a breast cell nuclei; these results will assist doctors in quickly identifying malignant and benign tumors in patients of breast cancer—an integral part of prioritizing treatment and its methods. This analysis ultimately allows for expedient diagnoses for the subsequent treatments of breast cancer, thereby benefiting millions of people all across the world.

## Data Dictionary

### Breast Cancer Risk Factor Dataset

Column/Feature Title	Data Type	Description
Year	Integer	Year of data in the row
Age_Group_5_Years	Integer	Age groups separated by every 5 years from 30-85+ as well as a group for 18-29
Race_Eth	Integer	Numerical categorized races (1=Non-Hispanic white, 2=Non-Hispanic black, 3=Asian/Pacific Islander, 4=Native American, 5=Hispanic, 6=Other/mixed, 9=Unknown)
First_Degree_HX	Boolean	Family history of breast cancer by first-degree relatives
Age_Menarche	Integer	Age at first occurrence of menstruation
Age_First_Birth	Integer	Age at first birth
BIRADS_Breast_Density	Integer	Level of breast tissue density from 1-4 in increasing densities
Current_HRT	Boolean	Whether or not the patient uses hormone replacement therapy
Menopause	Integer	Menopausal status (1=Pre or peri-menopausal, 2=Post-menopausal, 3=Surgical menopause)
BMI Group	Integer	Body mass index of the patient according to classification groups (1=10-25, 2=25-30, 3=30-35, 4=35+)
Biophx	Boolean	Whether or not the patient has had a previous breast biopsy or aspiration
Breast_Cancer_History	Boolean	Whether or not the patient has had a previous diagnosis of breast cancer
Count	Integer	The number of times this covariate has occurred in the dataset

## Quantitative Breast Cancer Cell Data

Column/Feature Title	Data Type	Description
Diagnosis	boolean	0 = Benign, 1 = Malignant
Radius	float64	Mean distance from center to each perimeter point ( $\mu\text{m}$ )
Texture	float64	Standard deviation of grey-scaled spots on cells
Perimeter	float64	Distance around the cell nucleus border ( $\mu\text{m}$ )
Area	float64	Measured area of the cell nucleus ( $\mu\text{m}^2$ )
Smoothness	float64	Variation in radius lengths
Compactness	float64	Formula calculated $\text{perimeter}^2/\text{area}$
Concavity	float64	Depth of concave portions of the cell contour
Concave Points	float64	Number of concave portions of the cell contour
Symmetry	float64	0-1 scale of the similarity between the left/right sides
Fractal Dimension	float64	Complexity of the contours/patterns in the breast tissue measured on a scale of 0-1

## Images of Breast Cancer Cell Data

Image Class	Data Type	Description
Class 0 (Benign)	composite	All images of Non-Cancerous IDC Breast Cancer Cells and scans
Class 1 (Malignant)	composite	All images of Cancerous IDC Breast Cancer Cells and scans

## Purpose

The average woman bears a 13% risk of developing breast cancer in her lifetime. (American Cancer Society, 2024). Last year, there were about 297,790 new cases of breast cancer in the United States alone—a staggering total that contributed to breast cancer as the most common type of cancer in American women (one in eight American women is at risk) (National Cancer Institute, 2020). On top of being a common type of cancer, breast cancer was the third leading cause of mortality burden (measured by years of life lost) in women in 2015, accounting for 6% of the mortality burden (Cancer Council Australia, 2023). Furthermore, breast cancer leads to significant hair loss, unwanted body modifications, and loss of fertility, all of which contribute to a profound impact on a patient's physical and emotional well-being, as well as their self-image and quality of life (Healthdirect, 2021). Healthcare officials across the world share a goal: to reduce cases and prevent or cure breast cancer in all (Breast Cancer Research Foundation, 2024). In response to this aspiration, we aim to implement a comprehensive model that addresses the numerous impacts of breast cancer on women. By analyzing the Breast Cancer cells we will diagnose if a patient has Breast Cancer through the analysis of key data points in the cells. This analysis aims to aid the treatment process from start to finish, targeting areas and demographics with high likelihood of developing breast cancer to diagnosing it through a neural-network/computer-vision model.

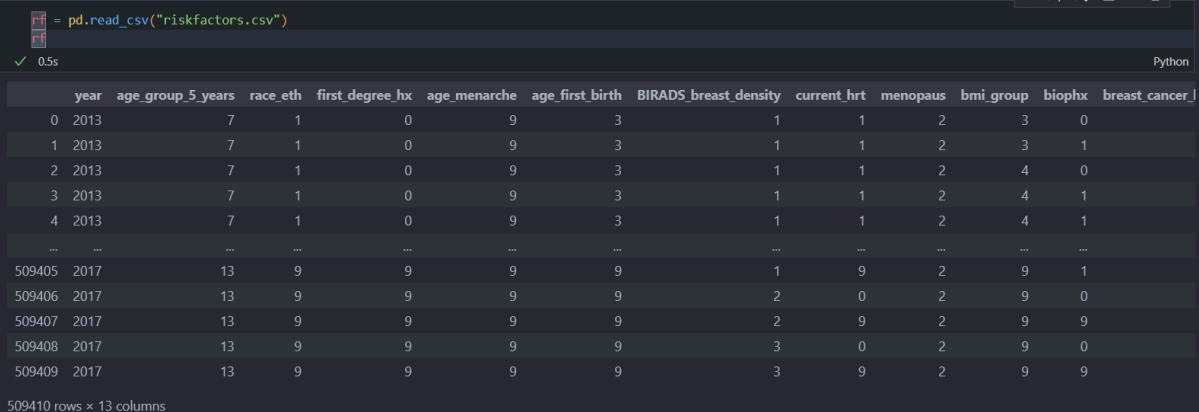
## Methods

1. [BCSC Research](#): The Breast Cancer Surveillance Consortium is a research network with breast cancer datasets whose risk factors dataset we used.
2. [UC Irvine Machine Learning Repository](#): We used UC Irvine's open-source database to sift through datasets concerning breast cancer and other health-related issues. We found a well-cited and recent dataset that had complete variables with a large amount of rows and columns.
3. [Correlation Matrix](#): Table of 0-1 coefficients between numerical variables
4. [K-Neighbors Classification](#): K-Nearest Neighbors (KNN) classification is a popular machine learning method that groups up the results of training data into different classifications based on their "locations" in respect to the independent variables. For each tested set of data, the algorithm looks at a select number of the closest trained data points and assigns a single point a binary classification depending on the dominant neighbor type.
5. [Confusion Matrix](#): A table that shows the number of true/false positives and negatives to assess the performance of a machine learning classification.
6. [ResNet](#): Residual Neural Networks use far fewer hidden layers in comparison to typical Neural Networks, employing a method called skip connections to bypass inaccurate bad layers by connecting a layer to an activation function much further down to increase accuracy in image classification.
7. [Precision-Recall Curve](#): A Precision-Recall Curve calculates the rate of having a correct positive prediction in comparison to all positive predictions versus correct positive predictions to false negative predictions.

# Results

## Exploratory Data Analysis

First, we imported our datasets and converted them into usable Pandas Data Frames in Python.



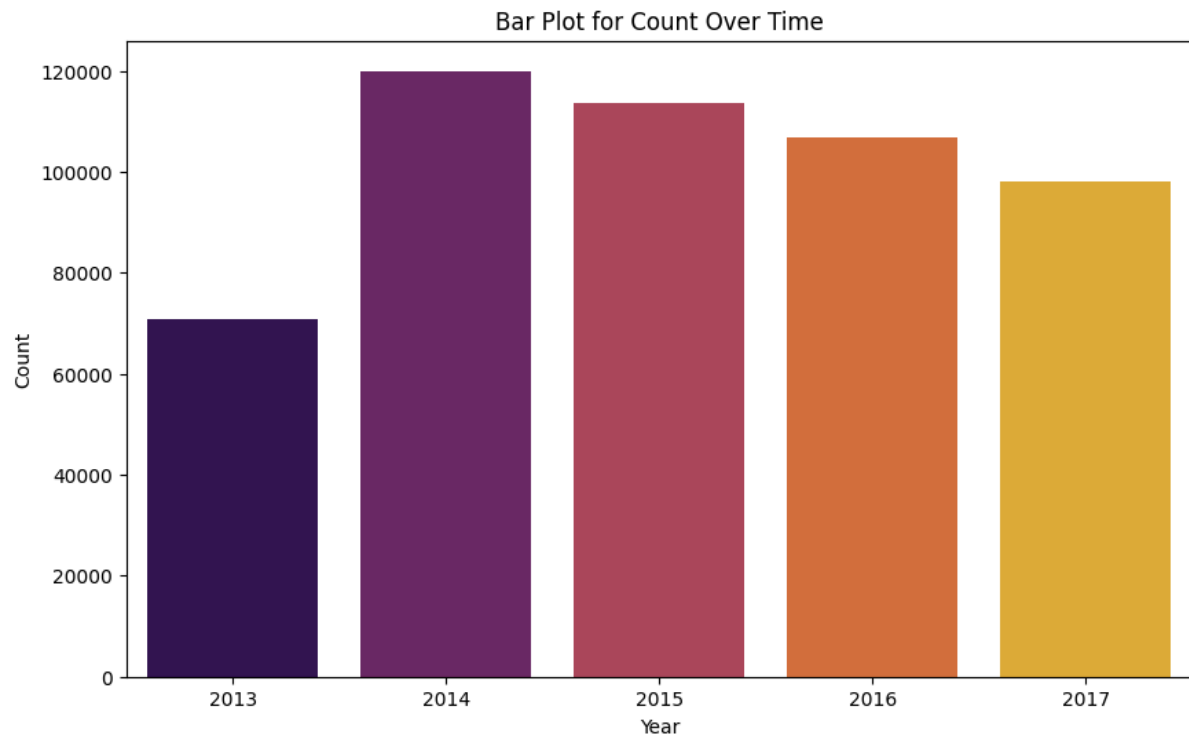
```
df = pd.read_csv("riskfactors.csv")
```

	year	age_group_5_years	race_eth	first_degree_hx	age_menarche	age_first_birth	BIRADS_breast_density	current_hrt	menopaus	bmi_group	biophx	breast_cancer_l
0	2013	7	1	0	9	3	1	1	2	3	0	
1	2013	7	1	0	9	3	1	1	2	3	1	
2	2013	7	1	0	9	3	1	1	2	4	0	
3	2013	7	1	0	9	3	1	1	2	4	1	
4	2013	7	1	0	9	3	1	1	2	4	1	
...	...	...	...	...	...	...	...	...	...	...	...	...
509405	2017	13	9	9	9	9	1	9	2	9	1	
509406	2017	13	9	9	9	9	2	0	2	9	0	
509407	2017	13	9	9	9	9	2	9	2	9	9	
509408	2017	13	9	9	9	9	3	0	2	9	0	
509409	2017	13	9	9	9	9	3	9	2	9	9	

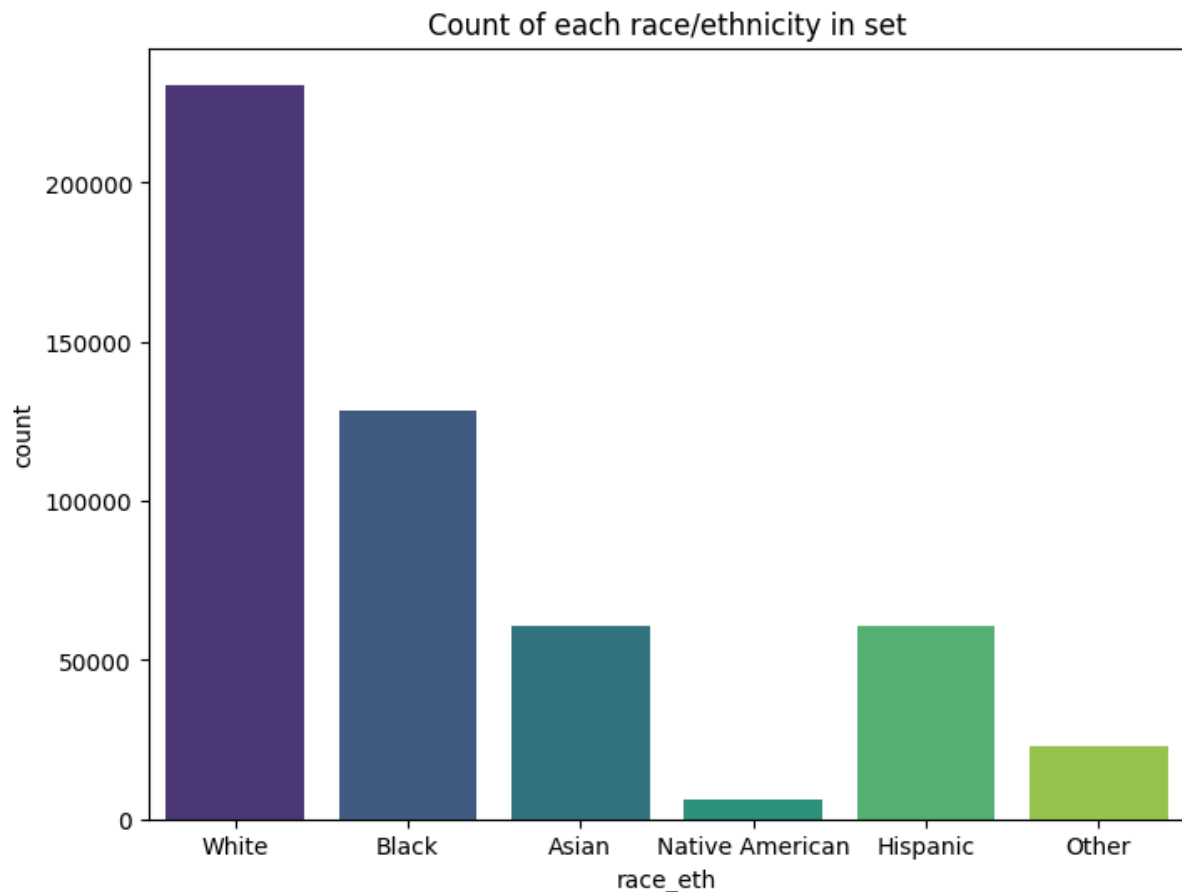
509410 rows x 13 columns

First, we had to clean this dataset. Every value that was a 9 represented an unknown data point. In order to remove these unknown values, we first converted all of the 9s into NaN (null) values, then we filled each of those null values with the mean of the remaining true values. These NaN values comprised about 6% of the true dataset, which is a small portion of 500,000 rows of information. Once the dataset was cleaned and the values were correct, we could conduct exploratory data analysis.

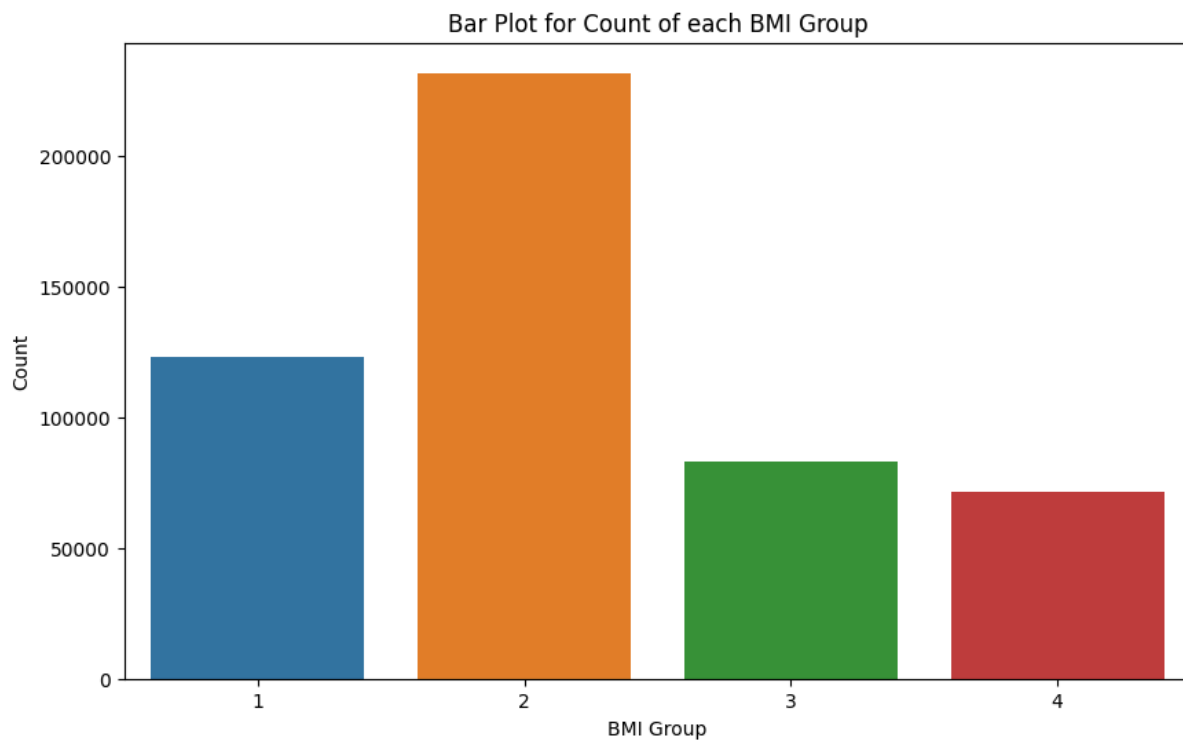




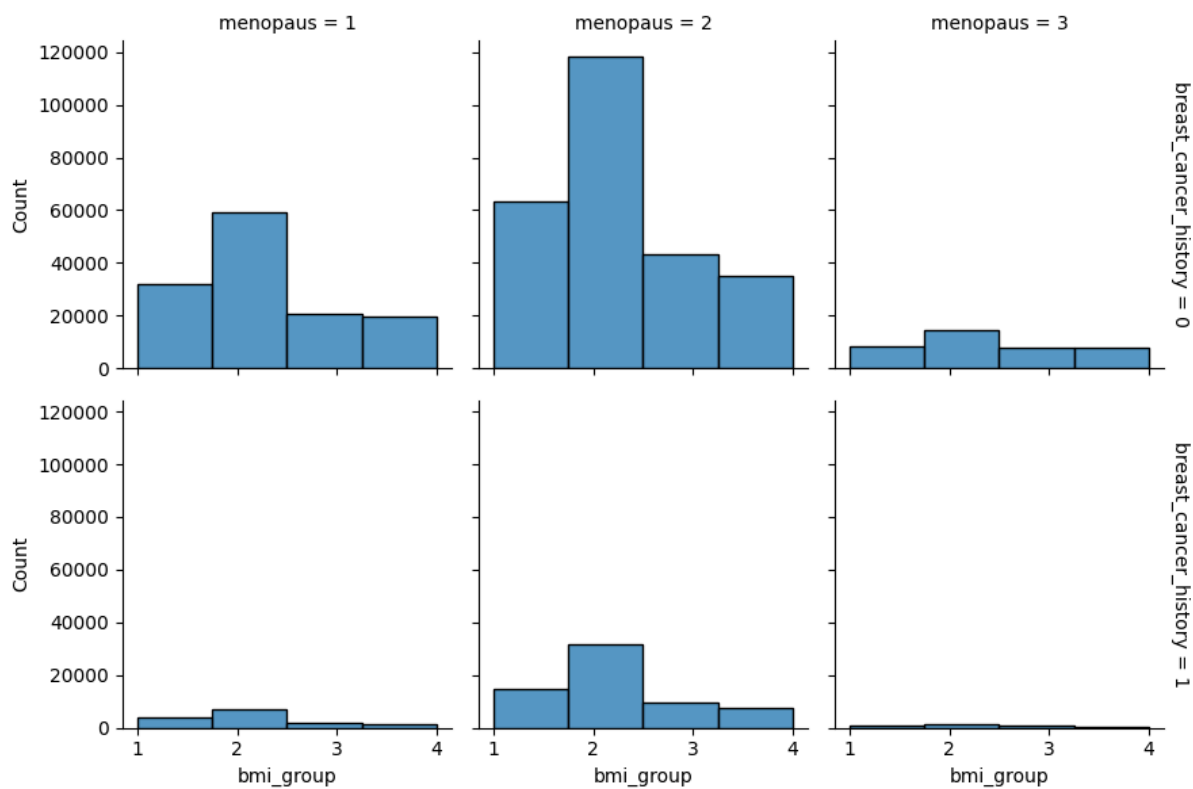
This scatter plot showcases a gradual decline in the Breast Cancer cases over the dataset following a spike from 2013-2014. However, if we look at current statistics, we can see that these numbers have actually increased over time and that breast cancer has re-emerged as a prominent and high-risk cancer in the world.



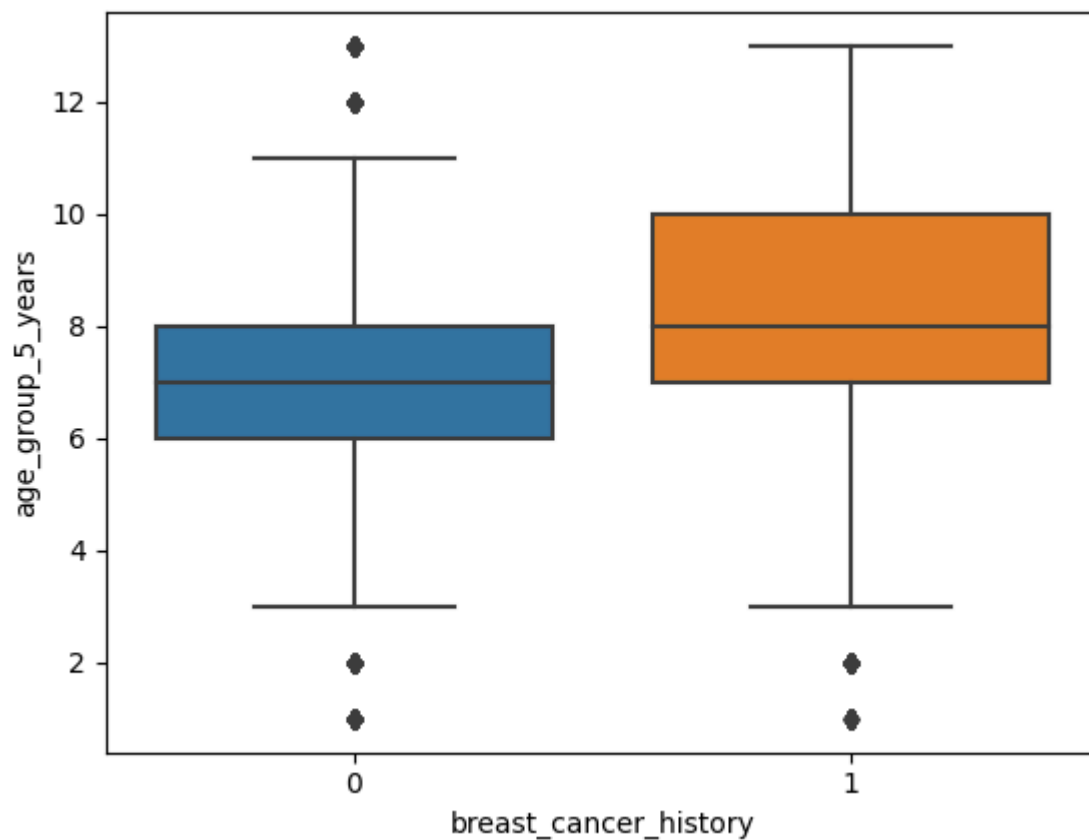
This bar graph showcases that the ethnicity in the set with the highest count of Breast Cancer cases is those of white descent, followed by black descent. These trends line up fairly well with the population trends in the US, however, white and black individuals are at a much higher risk for breast cancer.



This bar plot of the BMI groups shows that the BMI group of individuals between 25-30 BMI had nearly double the number of breast cancer cases of other groups. This means that the highest risk BMI group is those who are in the mid-tier BMIs.

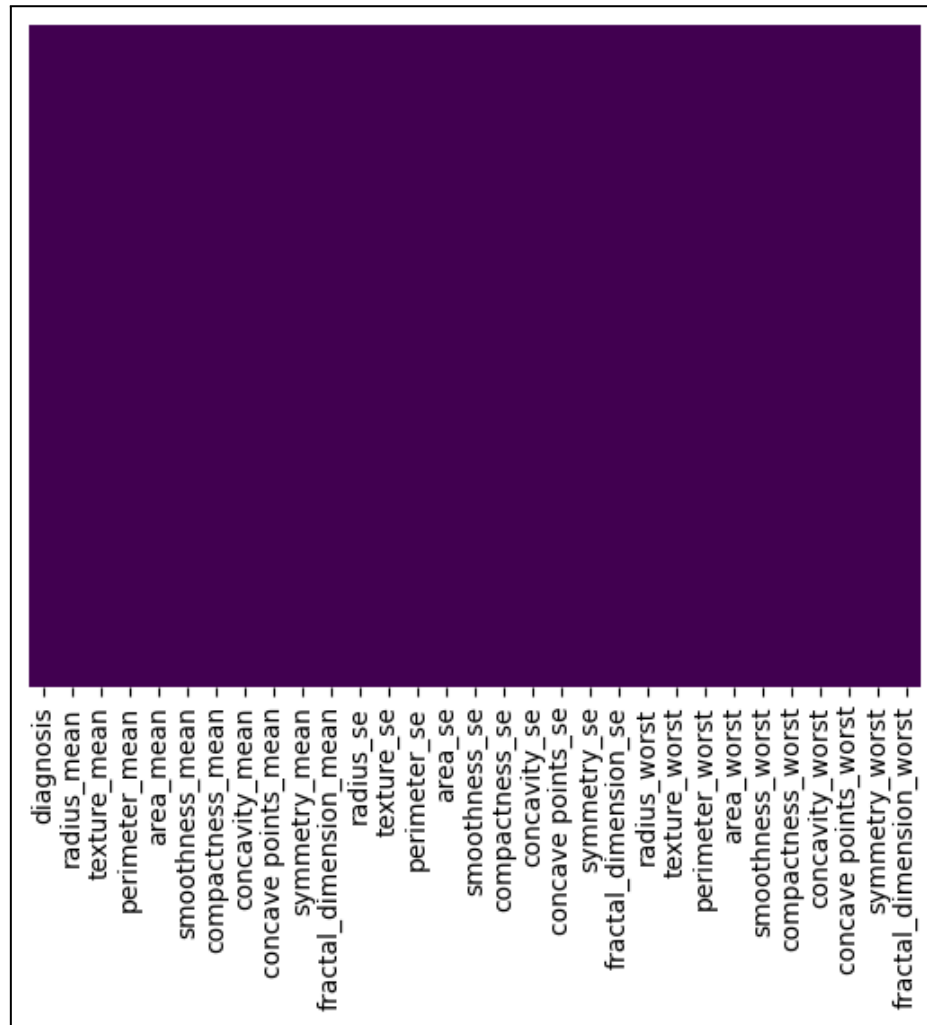


However, when we look closer at the relationship between BMI and Menopausal status, those in post-menopausal stages have a much higher risk at higher BMI groups than in any other higher BMI group category, whereas surgical menopausal women have a much lower risk for breast cancer in general across all BMI groups. This trend is interesting to note as BMI has a complicated relationship with breast cancer status.

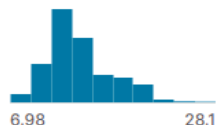
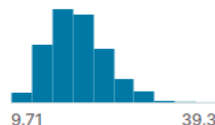
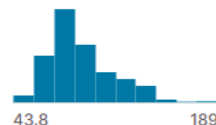
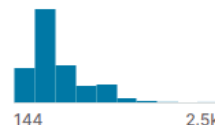


Lastly, we look at the relationship of the age groups and breast cancer history. Those who had a repeated diagnosis of breast cancer naturally were a lot older, but there is no observable statistical difference between the two. However, it can be deduced from the data that older women are at an increased risk of developing breast cancer, especially with a history of breast cancer diagnoses.

## Machine Learning (KNN) Classification Results



Then, we used the dataset from the UC Irvine Machine Learning Repository that contains information on different Breast Cancer cells. For each of the data columns, there is no missing data as seen through the Seaborn null-variable heatmap, which tests each variable to look for null values, which would show up as a yellow bar on the heatmap. Every independent variable is valid and can be considered in the analysis for now, discounting those removed through preliminary data cleaning.

diagnosis	# radius_mean	# texture_mean	# perimeter_mean	# area_mean
Target: M - Malignant B - Benign	Radius of Lobes	Mean of Surface Texture	Outer Perimeter of Lobes	Mean Area of Lobes
B 63%				
M 37%				
M	17.99	10.38	122.8	1001
M	20.57	17.77	132.9	1326
M	19.69	21.25	130	1203
M	11.42	20.38	77.58	386.1
M	20.29	14.34	135.1	1297
M	12.45	15.7	82.57	477.1
M	18.25	19.98	119.6	1040
M	13.71	20.83	90.2	577.9
M	13	21.82	87.5	519.8
M	12.46	24.04	83.97	475.9

The dataset contains information on nearly 600 breast cancer cells, detailing characteristics about the cells including the radius of lobes, mean of surface texture, outer perimeter of lobes and mean area of lobes.

```
df2 = df.drop(columns=["Unnamed: 32", "id"])
✓ 0.0s
```

In here we also realize there are 2 unnecessary columns, "id" and "Unnamed: 32" both of which are dropped from our dataset. The rest of these factors in our model are crucial to understanding the intricacies behind breast cancer cells and determining what features may aid our model in accurately diagnosing breast cancer in patients, thus facilitating a more efficient diagnostic process.

```

from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
df2['diagnosis'] = labelencoder.fit_transform(df2['diagnosis'].values)
df2['diagnosis'].value_counts()
✓ 0.0s
0    357
1    212
Name: diagnosis, dtype: int64

```

To start off by encoding our diagnosis with the binary values of 0 (Benign) and 1 (Malignant). We can see a spread of data that lines up well with real-life amounts of benign and malignant breast cancer cases.

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean
diagnosis	1.000000	0.730029	0.415185	0.742636	0.708984	0.358560	0.596534	0.696360	0.776614	0.330499	0.012838
radius_mean	0.730029	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	0.822529	0.147741	0.311631
texture_mean	0.415185	0.323782	1.000000	0.329533	0.321086	0.023389	0.236702	0.302418	0.293464	0.071401	0.076437
perimeter_mean	0.742636	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136	0.850977	0.183027	0.261477
area_mean	0.708984	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983	0.823269	0.151293	0.283110
smoothness_mean	0.358560	0.170581	0.023389	0.207278	0.177028	1.000000	0.659123	0.521984	0.553695	0.557775	0.584792
compactness_mean	0.596534	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0.831135	0.602641	0.565369
concavity_mean	0.696360	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.921391	0.500667	0.336783
concave points_mean	0.776614	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.000000	0.462497	0.166917
symmetry_mean	0.330499	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462497	1.000000	0.479921
fractal_dimension_mean	0.012838	0.311631	0.076437	0.261477	0.283110	0.584792	0.565369	0.336783	0.166917	0.479921	1.000000

We then create a correlation matrix with our variables and identify the variables that showcase the most association with diagnosis. Initially from looking at the correlation matrix, the means that correlate the most with the diagnosis are the number of concavities exhibited on the cell nucleus as well as size parameters like the radius and perimeter. However, the most important part of this correlation matrix is solely what correlates with the specific 'diagnosis' column, so we wanted to highlight just those in the next part.

```

diagnosis      1.000000
concave points_worst  0.793566
perimeter_worst  0.782914
concave points_mean  0.776614
radius_worst    0.776454
perimeter_mean  0.742636
area_worst      0.733825
radius_mean     0.730029
area_mean       0.708984
concavity_mean  0.696360
concavity_worst 0.659610
compactness_mean 0.596534
compactness_worst 0.590998
radius_se       0.567134
perimeter_se    0.556141
area_se         0.548236
texture_worst   0.456903
smoothness_worst 0.421465
symmetry_worst  0.416294
texture_mean    0.415185
concave points_se 0.408042
smoothness_mean 0.358560
symmetry_mean   0.330499
fractal_dimension_worst 0.323872
compactness_se  0.292999
...
smoothness_se   0.067016
fractal_dimension_mean 0.012838
texture_se       0.008303
symmetry_se      0.006522
dtype: float64

```

This sorting allows us to easily analyze our data and notice the necessary correlations between the independent and dependent variables. Here we find the variables “concave point worst”, “perimeter worst”, “concave point mean” and “radius



worst" show the most association with our diagnosis variable, which stays in line with our knowledge of the size and concavity being the two biggest factors.

We now dive into building our K-Nearest Neighbor Machine Learning Classification. The K-Nearest Neighbor Machine Learning Classification is a simple, but powerful machine learning algorithm used for classification and regression tasks. The machine learning algorithm classifies new observations based on the majority class of its "k" closest training examples, in our project KNN will be used to find closest neighbors of individual cells and utilize the characteristics of the cells to determine if the cell is Benign or Malignant.

```
x = df2.drop('diagnosis', axis=1).values
y = df2['diagnosis']
✓ 0.0s
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x = scaler.fit_transform(x)
```

To begin our KNN model, we exported our diagnosis variable to a 'Y' output column and dropped it from our features data frame. We then utilize `StandardScaler()` to standardize the dataset's features ensuring each has a mean of zero and a standard deviation of one. This process is fundamental to make sure features with large numerical ranges dominate the distance computation between data points, leading to a biased outcome. This guarantees all of the features are on a level playing field, thereby contributing equally to the calculation of distances between data points.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=12)
```

✓ 0.0s

To assess our model's predictive capability, we partition the dataset into a training set and testing set, adhering to a 70-30 split. The division allows our model to learn through the training data and be evaluated through the testing dataset. The approach allows us to effectively gauge the model's generalization ability and its performance in real world scenarios, ensuring reliability and robustness in predictions.

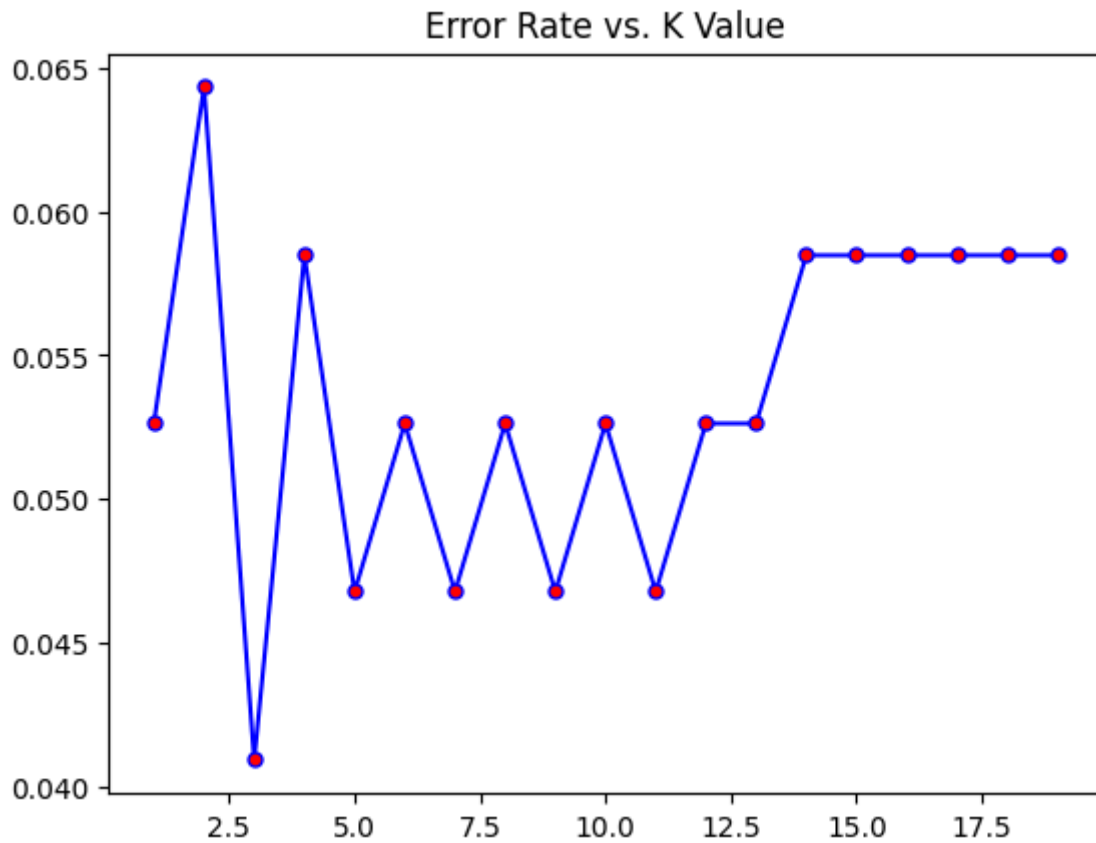
```
model = KNeighborsClassifier(n_neighbors = 1)
model.fit(X_train, y_train)
predicted = model.predict(X_test)

print(confusion_matrix(y_test, predicted))
print("Accuracy (KNN[1]): ", accuracy_score(y_test, predicted.round())*100, "%")
```

✓ 0.0s

```
[[104  3]
 [ 6 58]]
Accuracy (KNN[1]): 94.73684210526315 %
```

The core of this portion of our project involves training a K-Nearest Neighbors classifier using the standardized testing data. After the training phase the model undergoes a comprehensive evaluation on the test set. We employ a suite of metrics for these evaluations, notably a confusion metrics and accuracy score, which collectively offers insight into the model's predictive accuracy and its ability to discern between benign and malignant cells with precision.



An integral aspect of the KNN model involves optimizing the number of nearest neighbors ( $k$ ), a parameter that significantly influences the model's predictive performance. Through an interactive process, we plot the error rate against varying values of  $k$  to identify the optimal number that minimizes the error rate. This optimization ensures that our model achieves the best possible balance between sensitivity and specificity, therefore enhancing its diagnostic utility in identifying the nature of cellular samples. Our optimal number of neighbors was 3.

```

model = KNeighborsClassifier(n_neighbors = 3)

model.fit(X_train, y_train)
predicted = model.predict(X_test)

print('Confusion Matrix :')
print(confusion_matrix(y_test, predicted))

print()
print("Accuracy (KNN[5]): ", accuracy_score(y_test, predicted.round())*100, "%")

```

✓ 0.0s

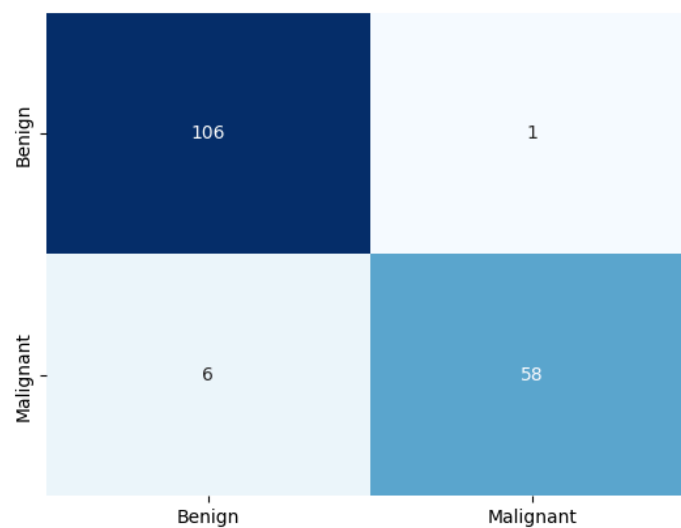
Confusion Matrix :

```

[[106  1]
 [ 6 58]]

```

Accuracy (KNN[5]): 95.90643274853801 %



After we plotted our error rate against varying values of  $k$ , we came up with the conclusion that the value of 3 neighbors provided us with the best results. The final accuracy of our model through our `accuracy_score` and confusion matrix was 95.9%.

## Computer Vision (ResNet) Cleaning

```
tfm = transforms.Compose([
    transforms.Resize((224, 224)), #standard size of images for resnet
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(degrees = 10),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]) #normalize pictures to fit imagenet format
])
```

Data Engineering Algorithm #1: Reformatting the images. By utilizing the torchvision library we are able to efficiently create a transformation function that runs an array of operations on whatever images the function takes in. In this case, the first 3 transformations handle the physical transformations of the images in order to fit them into the correct format input format for the ResNet model. We set the size of the image to 224 x 224, flip it horizontally and use random rotation to prevent the model from overfitting onto a specific composition of images. Finally, the ToTensor command converts an image with dimensions (Height x Weight x Color Channel) to the correct (Color Channel x Height x Weight) format.

```
tfm = transforms.Compose([
    transforms.Resize((224, 224)), #standard size of images for resnet
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(degrees = 10),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]) #normalize pictures to fit imagenet format
])
```

Data Engineering Algorithm #2: Normalization. Using the normalization function we are able to center and scale the pixel values to fit the input requirements for the ResNet model. The specific means and standard deviations to normalize to are given here. Normalization, in addition to the ResNet model itself, ensures that the gradients during backpropagation are well-scaled and don't cause issues like vanishing or exploding gradients.

## Computer Vision (ResNet) Results

The computer vision implementation revolved heavily around utilizing the prebuilt ResNet 18 model. After that we adjusted the final, fully connected layer, to fit our specific needs of binary classification. The code for those two processes are seen below.

```
model = models.resnet18(weights = ResNet18_Weights.DEFAULT)
#replace the final (fully connected) layer with what we want
#input features remain the same
model.fc = Linear(in_features = 512, out_features = 2) #we have 2 outputs
```

After establishing our optimizer (Adam) and our loss function (Cross Entropy) we were able to follow the generic implementation of the training process which included running a tqdm (progress bar) as well as calculating numerous statistics to give information about how the model was performing after each epoch (train accuracy, loss calculations, backpropagation, minutes per epoch).

```
with tqdm(train_loader, unit="batch") as tepoch:
    for xtrain, ytrain in tepoch: #for every iteration in train datasets
        optimizer.zero_grad()

        xtrain = xtrain.to(device) #both model and input need to be on the same device
        ytrain = ytrain.to(device)

        train_prob = model(xtrain) #output of model based
        train_prob = train_prob.cpu() #prevent maxing out gpu memory

        #calculate loss
        loss = loss_fn(train_prob, ytrain) #compare model output to actual output
        loss.backward() #backpropagation
        optimizer.step() #update model parameters (weights)

        running_loss += loss.item() * xtrain.size(0)
        # training ends

        train_pred = torch.max(train_prob, 1).indices #converts from probabilities to 1 and 0 (true and false)
        tr_acc += int(torch.sum(train_pred == ytrain)) #adds number of trues

    epoch_loss = running_loss / len(train_loader.dataset)
    ep_tr_acc = tr_acc / len(train_loader.dataset)

    train_losses.append(epoch_loss)
    train accuracies.append(ep_tr_acc)
```

Finally to evaluate our model using the test dataset, we utilized another for loop as well as the “with torch.no\_grad()” command to be sure to not update the gradients

during the evaluation process. In the evaluation stage of training, we were aiming to calculate the epoch test accuracy.

```
#Evaluate
model.eval()
with torch.no_grad(): # don't update gradients
    for xtest, ytest in test_loader:
        xtest = xtest.to(device)
        ytest = ytest.to(device)

        test_prob = model(xtest)
        test_prob = test_prob.cpu()

        test_pred = torch.max(test_prob, 1).indices
        test_acc += int(torch.sum(test_pred == ytest).item())

    ep_test_acc = test_acc / LEN_TEST
    val accuracies.append(ep_test_acc)
```

The results on three epochs were as follows.

```
100% |████████████████████████████████████████| 31/31 [01:03<00:00, 2.06s/batch]
Epoch: 0, Time: 1.3431390841801962, Loss: 0.0010608454467728734
Train_acc: 0.9848648648648649, Test_acc: 0.773071104387292

100% |████████████████████████████████████████| 31/31 [01:03<00:00, 2.03s/batch]
Epoch: 1, Time: 1.3431575854619344, Loss: 0.030036725103855133
Train_acc: 0.987027027027027, Test_acc: 0.8502269288956127

100% |████████████████████████████████████████| 31/31 [01:03<00:00, 2.03s/batch]
Epoch: 2, Time: 1.3412885149319966, Loss: 0.012844787910580635
Train_acc: 0.9848648648648649, Test_acc: 0.859304084720121
```

To showcase the results of our computer vision model we developed three visual representations of our results, a confusion matrix, training and validation accuracy graphs and finally a precision-recall curve.

```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(train_losses, label='Training Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(train_accuracies, label='Training Accuracy')
plt.plot(val_accuracies, label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training & Validation Accuracy')
plt.legend()
plt.show()
```

```

#confusion matrix
all_labels = []
all_preds = []
with torch.no_grad():
    for images, labels in test_loader:
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        _, preds = torch.max(outputs, 1)
        all_labels.extend(labels.cpu().numpy())
        all_preds.extend(preds.cpu().numpy())

conf_matrix = confusion_matrix(all_labels, all_preds)
plt.figure(figsize=(7, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=train_ds.classes, yticklabels=train_ds.classes)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

```

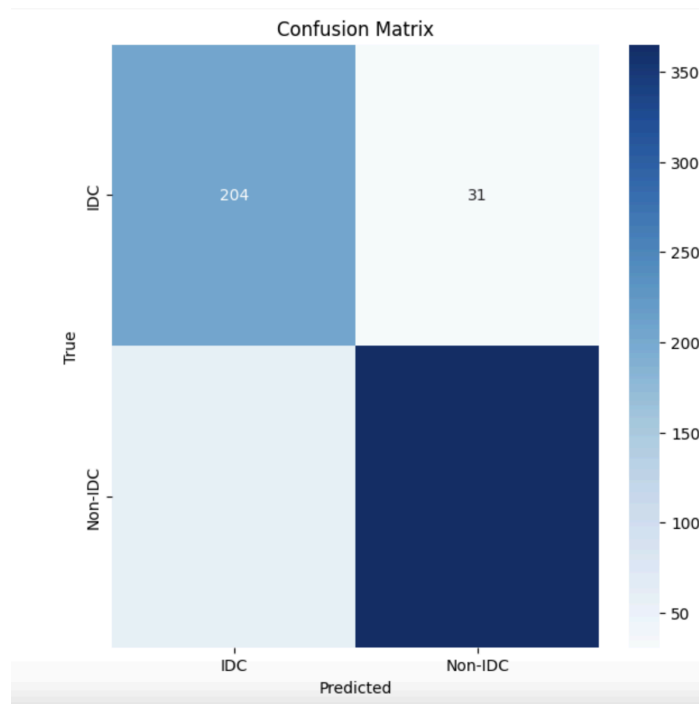
```

#Precision-Recall Curve
probs = []
true_labels = []
with torch.no_grad():
    for images, labels in test_loader:
        images = images.to(device)
        outputs = model(images)
        probabilities = torch.softmax(outputs, dim=1)[: , 1]
        probs.extend(probabilities.cpu().numpy())
        true_labels.extend(labels.cpu().numpy())

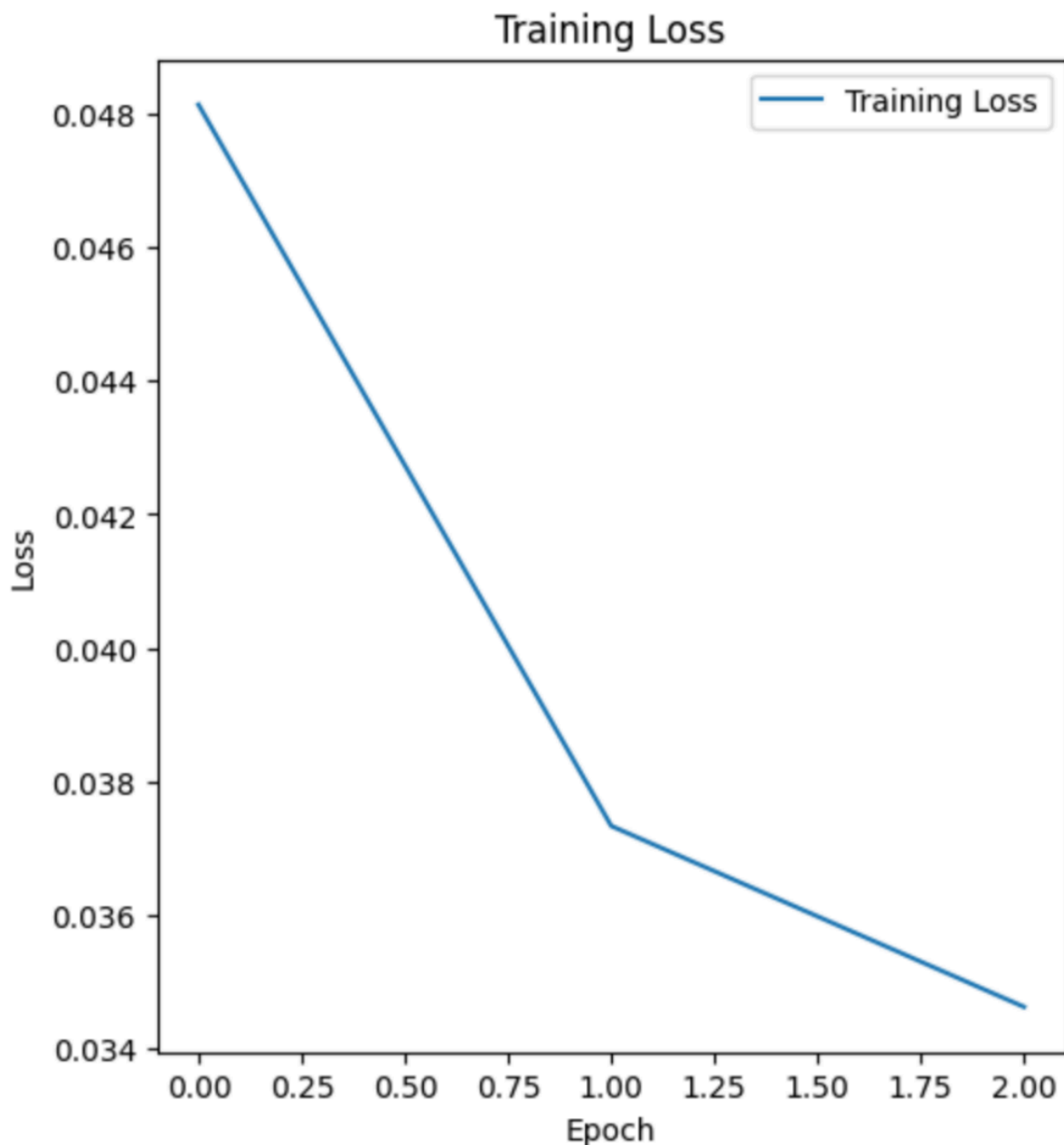
precision, recall, _ = precision_recall_curve(true_labels, probs)
plt.figure(figsize=(7, 5))
display = PrecisionRecallDisplay(precision=precision, recall=recall)
display.plot()
plt.title('Precision-Recall Curve')
plt.show()

```

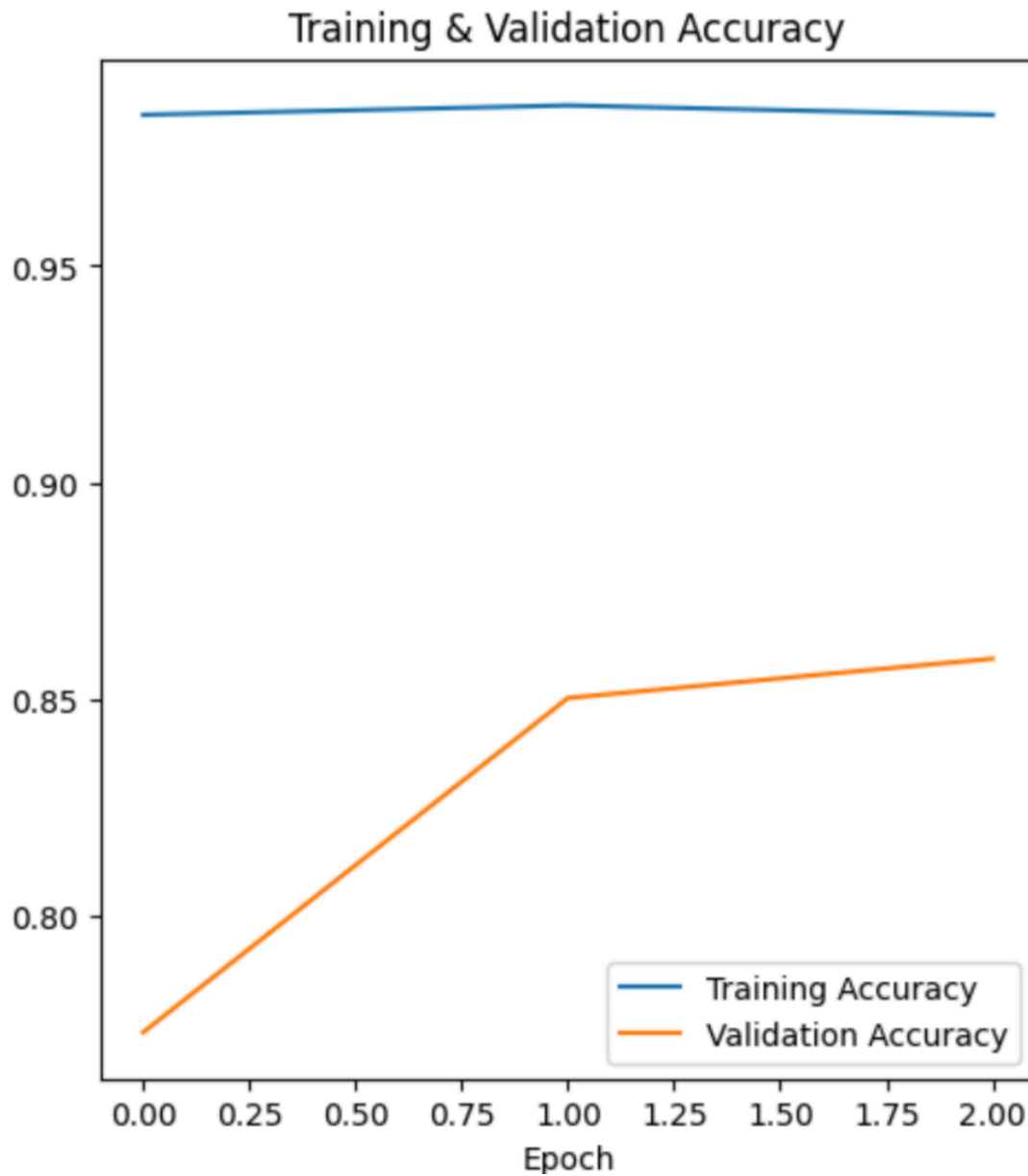




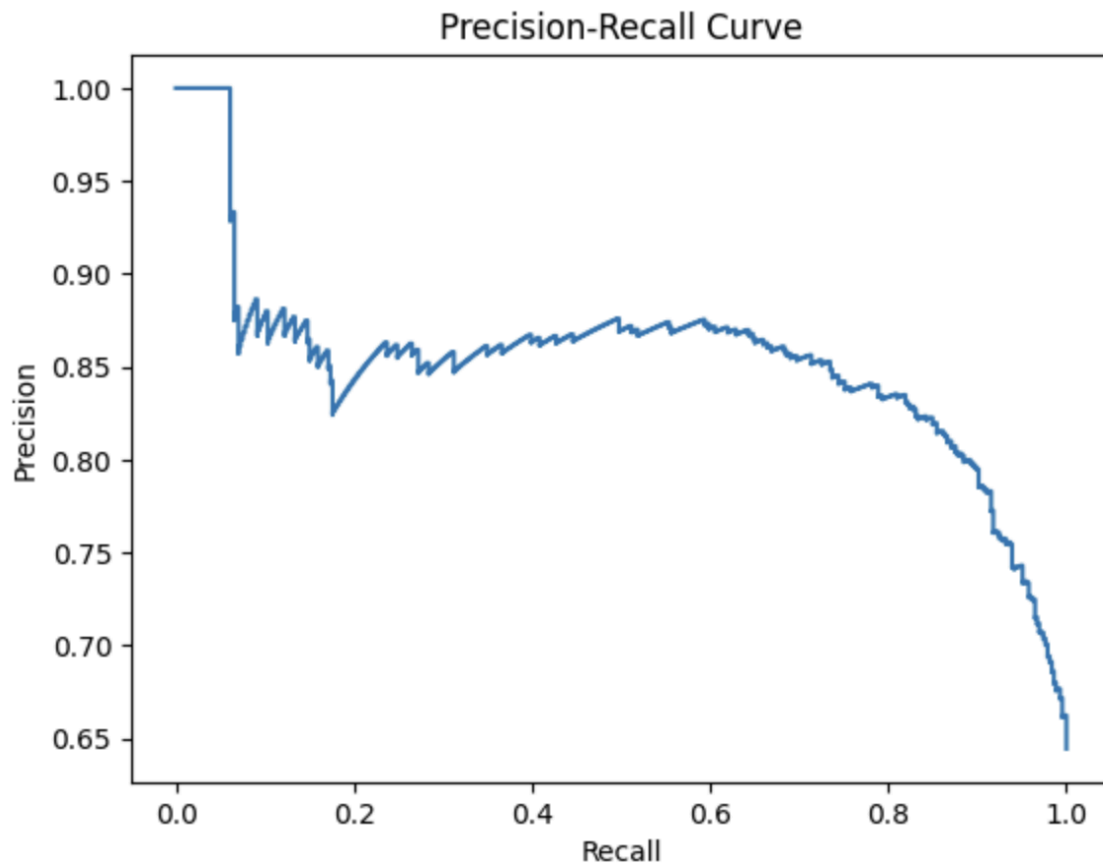
The confusion matrix indicates that our model has a strong predictive ability, successfully identifying 204 true positive IDC cases. However, it also shows room for improvement, with 31 false positives where Non-IDC cases were misclassified as IDC. Despite this, the relatively low number of false positives in comparison to true positives suggests that the model is quite reliable in detecting IDC.



The training loss curve reveals an encouraging trend of decreasing loss, implying that our model is effectively learning from the training data. The initial sharp decrease indicates rapid learning, which stabilizes over time, a pattern typical of well-functioning models that are converging towards optimal performance.



The accuracy curves demonstrate that the model achieves a commendable level of accuracy, approximately 87%, both in training and validation. This consistency between training and validation accuracy is a strong indicator of the model's robustness and its ability to generalize well to new, unseen data.



The precision-recall curve for our classification model starts with high precision at lower recall levels, indicating that initial predictions are highly accurate. As recall increases, capturing more true positives, precision gradually declines, reflecting the trade-off inherent to the model's prediction strategy. The curve suggests that the model maintains reasonable precision up to a certain point, even as it strives to identify most positive cases, which is indicative of its overall effectiveness in balancing the need for accurate and comprehensive positive case identification.

## Conclusion

The ResNet image classification highlighted where Breast Cancer cells have areas in which the signs of whether the cells are benign or malignant become apparent. This holistic data analysis approach enabled us to look at every aspect of a cell, both qualitatively and quantitatively, and correspond that with the metadata to provide meaningful, highly accurate results for breast cancer treatments. In order to achieve our goal of producing a computer vision model that efficiently identifies, hospitals and breast cancer practitioners need to focus their efforts on predictive analytics such as this. The 95% predictive accuracy in the K-Nearest Neighbors goes a step further with these results to illustrate the promise that such analysis holds. Being able to predict whether or not a given cell, based on the nucleus' dimensions and a mammogram of the cell, is cancerous is crucial to the development of improved breast cancer treatment technologies. Additionally, a slight correlation between age and breast cancer diagnoses augments a targeted population where breast cancer treatments are necessary to go to first; the older populations and those with history and undergoing hormone replacement therapy (HRT) are the highest at risk of developing breast cancer. Lastly, the complex BMI relationship and the remaining results all point toward the expected correlations that we saw in the data. These insights add to the validity of the computer vision model by specifying where the results of this data analysis should be applied more in-depth. Our analysis can be most beneficial to understanding breast cancer across America and to helping healthcare pursue its goals of decreasing breast cancer cases from the large amounts seen today, thus achieving the world's goal of reducing the number of cases.

## Next Steps

The speed of diagnosis of Breast Cancer is crucial to a patient's safety and recovery. With computer-vision-aided techniques, previously unidentifiable tumors can be classified and taken into account for future medical procedures. It's crucial for these machine learning techniques outlined in the project to be first tested on a wider range of breast cancer data detailing more patients, more niche cases, and a larger variety of breast cancer variants. After thoroughly combing through all of the possibilities, another step would be to distribute the combined system out to hospitals en masse. With data on other clinical variables taken without the help of computer vision, there is still a wide range of unaccounted-for indicators and causes of breast cancer that would provide a more accurate prognosis for trained doctors. Inputting such information into the machine learning model could potentially help more than just identifying breast cancer and its malignant or benign diagnosis but also introduce a long-term solution for individual treatment. Trends analyzed could include survival expectation over a long term and a timeline of tumor progression between the different stages, all through micro-analyses of the different factors and where one stage gives way to the next. With enough advancement, it could even be possible to weigh the benefits and risks of different methods of treatment, evaluating the efficacy of chemotherapy as to other strategies and whether to proceed treatment in certain situations where the cancer has evolved out of control. This future data analysis will have to balance the ethical concerns of using patient data combined with advancing technologies to push toward the goal of using a computer to mimic a doctor constantly monitoring the patient in question, to make accurate observations, and to determine the best path forwards off of images and small changes in data.

## References

American Cancer Society. (2024, January 17). Breast cancer.

<https://www.cancer.org/cancer/types/breast-cancer/about/how-common-is-breast-cancer.html>. Accessed 16 Feb. 2024.

Breast Cancer Research Foundation. (n.d.). Home. <https://www.bcrf.org/>. Accessed 16 Feb. 2024.

Breast Cancer Surveillance Consortium. (n.d.). Risk factor dataset download.

<https://www.bcscc-research.org/datasets/rf/risk-factor-dataset-download>. Accessed 16 Feb. 2024

Cancer Council Australia. (2023, September). Breast cancer - National Cancer Control Policy. [https://wiki.cancer.org.au/policy/Breast\\_cancer](https://wiki.cancer.org.au/policy/Breast_cancer). Accessed 16 Feb. 2024.

Healthdirect. (2023, October 5). Living with breast cancer.

<https://www.healthdirect.gov.au/living-with-breast-cancer>. Accessed 16 Feb. 2024.

Janowczyk, A., & Madabhushi, A. (n.d.). Deep Learning for Digital Pathology Image

Analysis: A comprehensive tutorial with selected use cases. Journal of pathology informatics. <https://pubmed.ncbi.nlm.nih.gov/27563488/>. Accessed 16 Feb. 2024.

National Cancer Institute. (2020, December 16). Breast cancer—Patient version.

<https://www.cancer.gov/types/breast/risk-fact-sheet>. Accessed 16 Feb. 2024.

Torchvision. (n.d.). Source code for torchvision.models.resnet.

torchvision.models.resnet - Torchvision 0.17 documentation.

[https://pytorch.org/vision/stable/\\_modules/torchvision/models/resnet.html](https://pytorch.org/vision/stable/_modules/torchvision/models/resnet.html).

Accessed 16 Feb. 2024.

University of California, Irvine. (n.d.). Breast Cancer Wisconsin (Diagnostic) [Data set].

UCI Machine Learning Repository.

<https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic>.

Accessed 16 Feb. 2024.

WCRF International. (2022, April 14). Breast cancer statistics. World Cancer Research Fund International.

<https://www.wcrf.org/cancer-trends/breast-cancer-statistics>. Accessed 16 Feb. 2024.

Wolberg, W., Mangasarian, O., Street, N., & Street, W. (1995). Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository.

<https://doi.org/10.24432/C5DW2B>. Accessed 16 Feb. 2024.

[All code—produced by the researchers—can be accessed at this Github Repository.](#)