CLASSIFICATION:

BREAST CANCER DETECTION

REPRESENTED BY: SAOUABEDDINE YOUNES

GUIDED BY BY: KHAMJANE AZIZ

CONTENT

INTORDUCTION

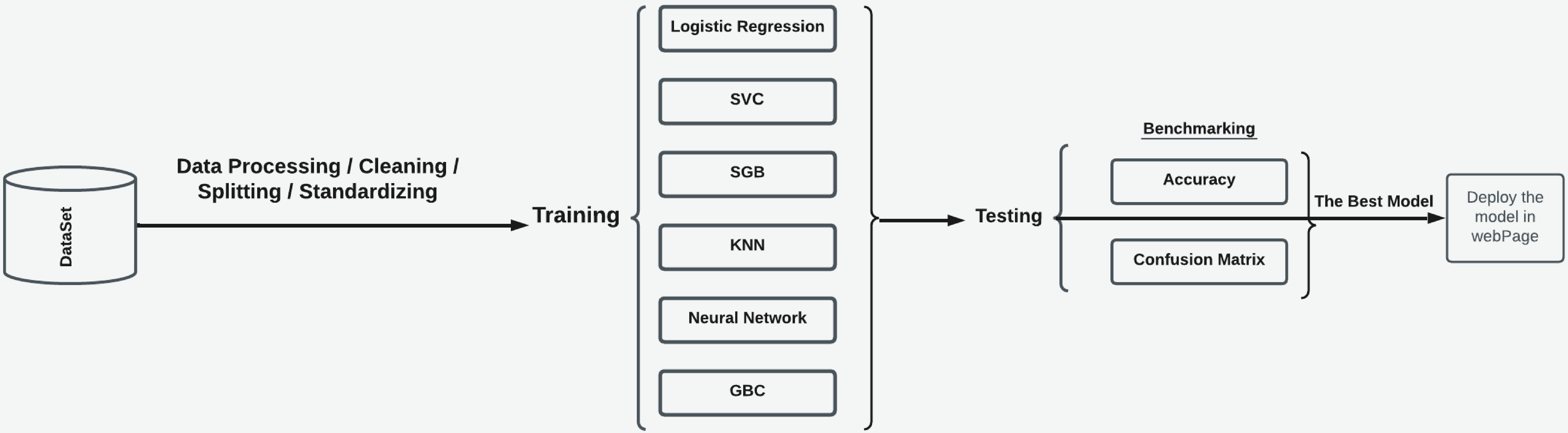
Breast cancer is a significant public health concern worldwide, with a high incidence rate among women. Early detection of breast cancer plays a crucial role in improving patient outcomes and survival rates.

Manual detection methods, such as physical examination and palpation, may have limited sensitivity (ability to detect true positive cases) and specificity (ability to avoid false positive cases).

“and that is what we are looking for in this project is to increase sensitivity and low rate of FP cases”

(569, 32)

ARCHITECTURE



DATA PROCESSING

```python
# Importing the required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
data = pd.read_csv("breast-cancer.csv")
```

```python
data.head()
```

Visualize

	id int64	diagnosis object	radius_mean float64	texture_mean float...	perimeter_mean fl...	area_mean float64	smoothness_mean f...	c
0	842302	M	17.99	10.38	122.8	1001	0.1184	
1	842517	M	20.57	17.77	132.9	1326	0.08474	
2	84300903	M	19.69	21.25	130	1203	0.1096	
3	84348301	M	11.42	20.38	77.58	386.1	0.1425	
4	84358402	M	20.29	14.34	135.1	1297	0.1003	

5 rows, showing 10 per page

Page 1 of 1

Data features

```
data.columns
```

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

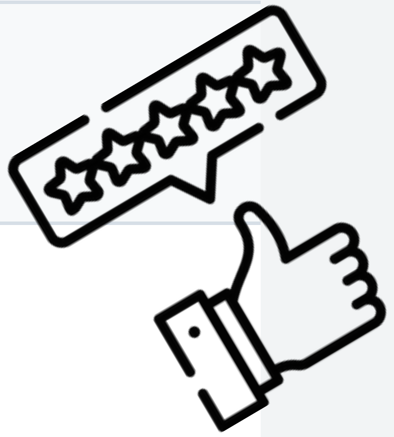
- We can notice that we have a feature called id, which is not important in our study so we will remove it soon.
- Diagnosis is a categorical variable.

```
data.drop('id', axis=1, inplace= True)

```

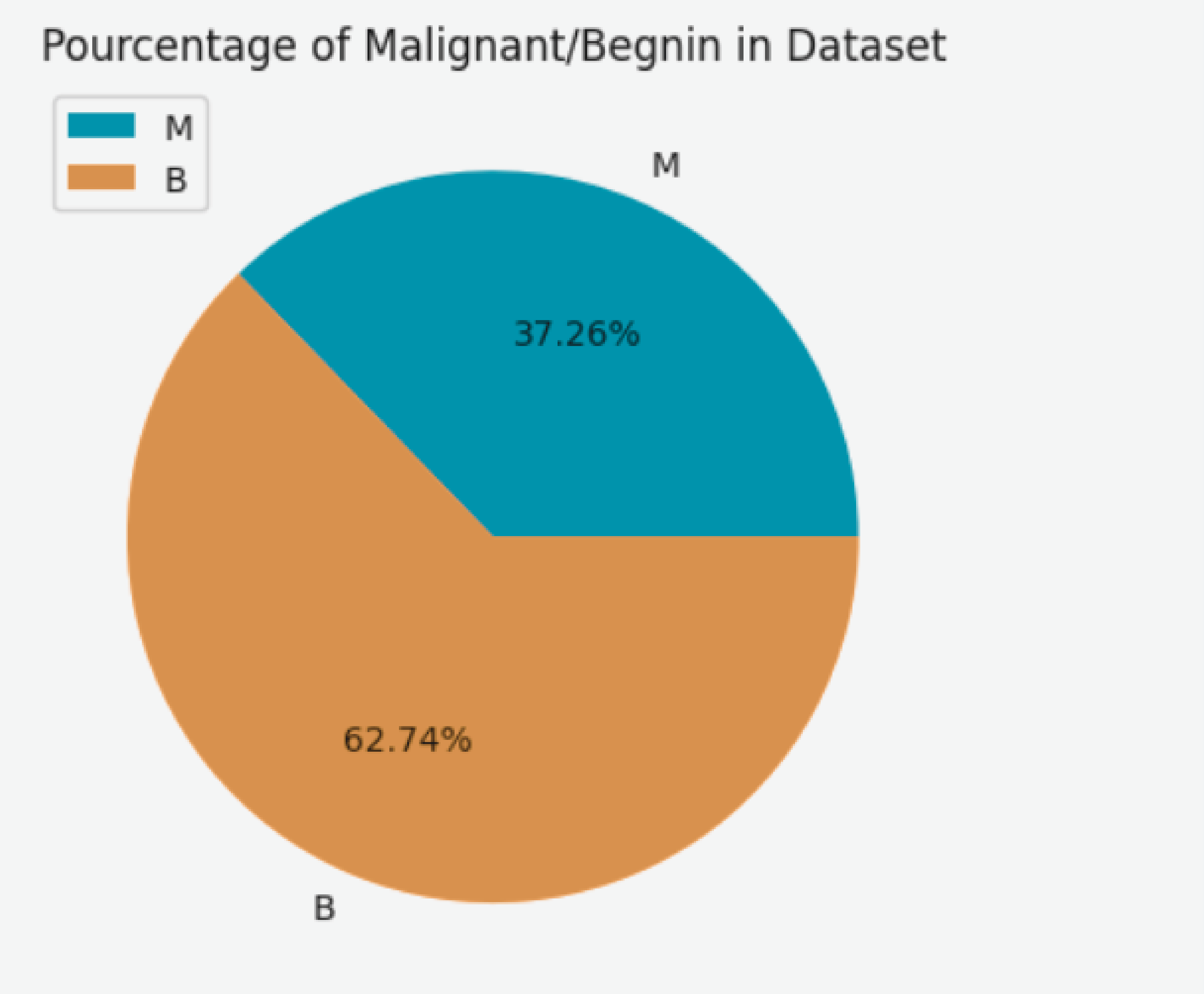
DATA VISUALIZATION

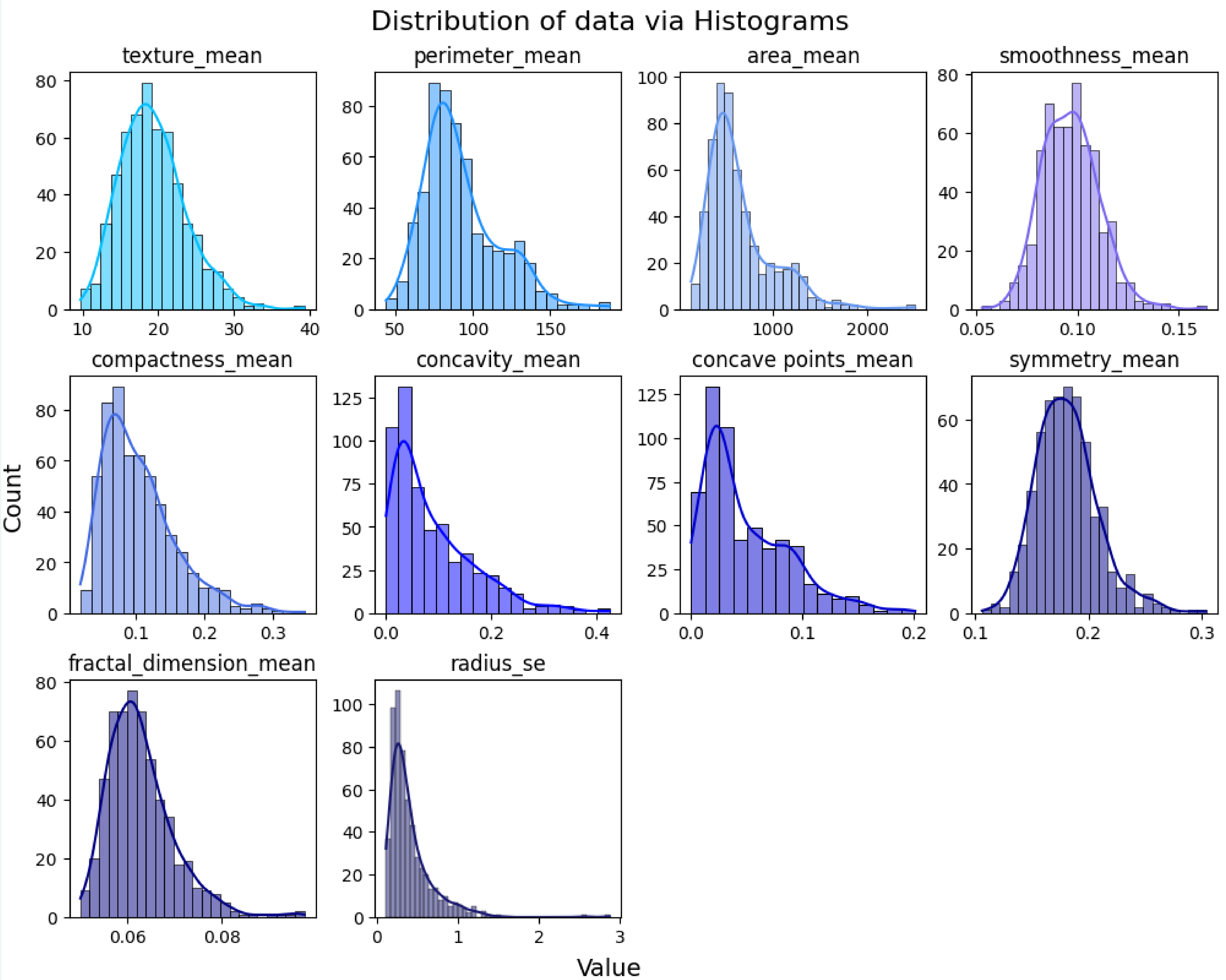
```python
data.isnull().sum().sum()
```

```
0
```

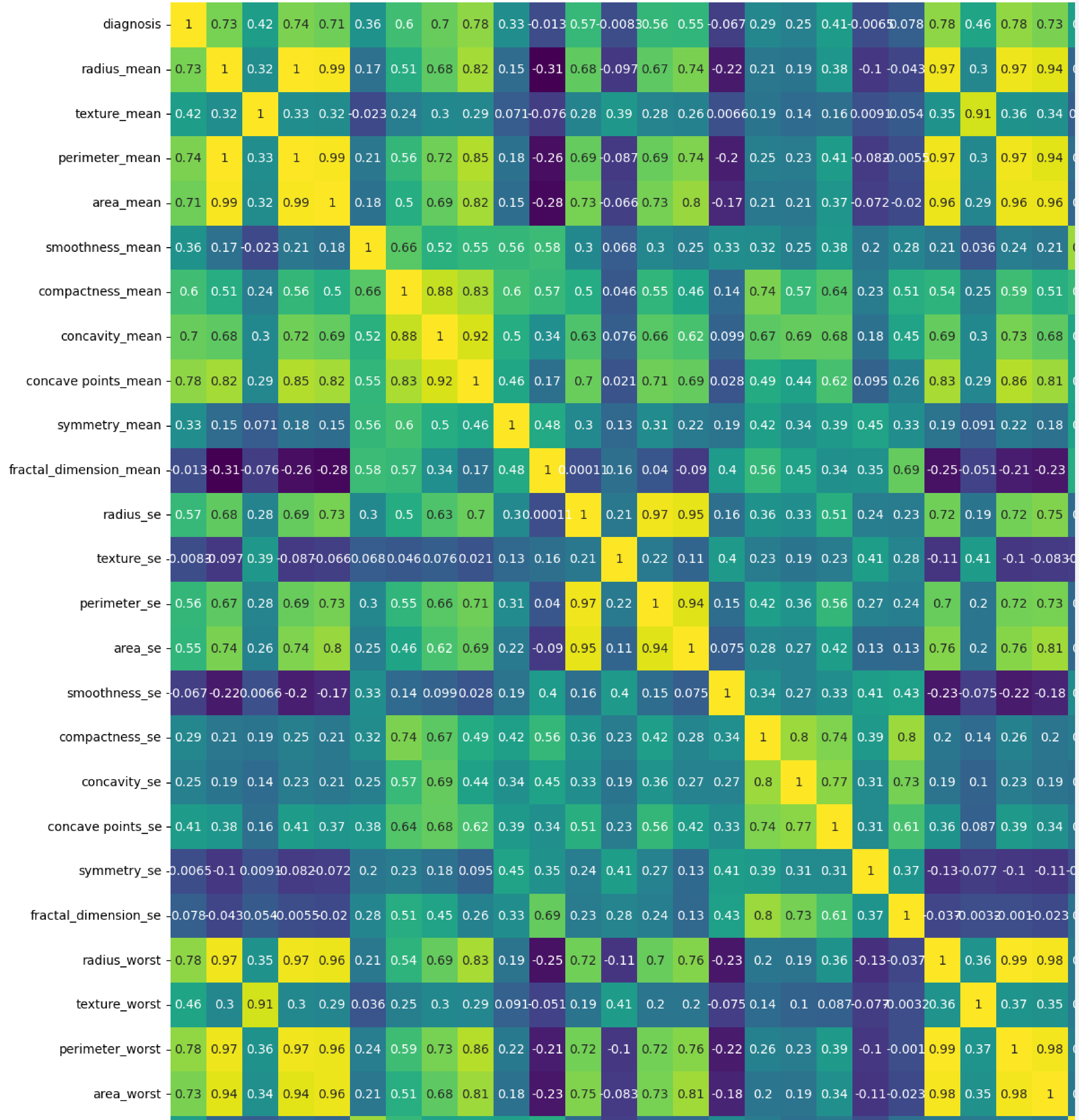


- As we can see, there is no NULL values in our data.





Distribution of data via Histograms
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
radius_se
Count
Value



CORRELATION MATRIX

'A lot of projects preserve only the features with correlation with >0.6 or >0.5 ,but it's not worth it'

DATA SPLITTING

```
X = data.drop(['diagnosis'], axis=1)
y = data['diagnosis']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
```

DATA STANDARDIZATION

$$x' = \frac{x - \bar{x}}{\sigma}$$

```
# scaling data
from sklearn.preprocessing import StandardScaler
s = StandardScaler()

X_train = s.fit_transform(X_train)
X_test  = s.fit_transform(X_test)
```

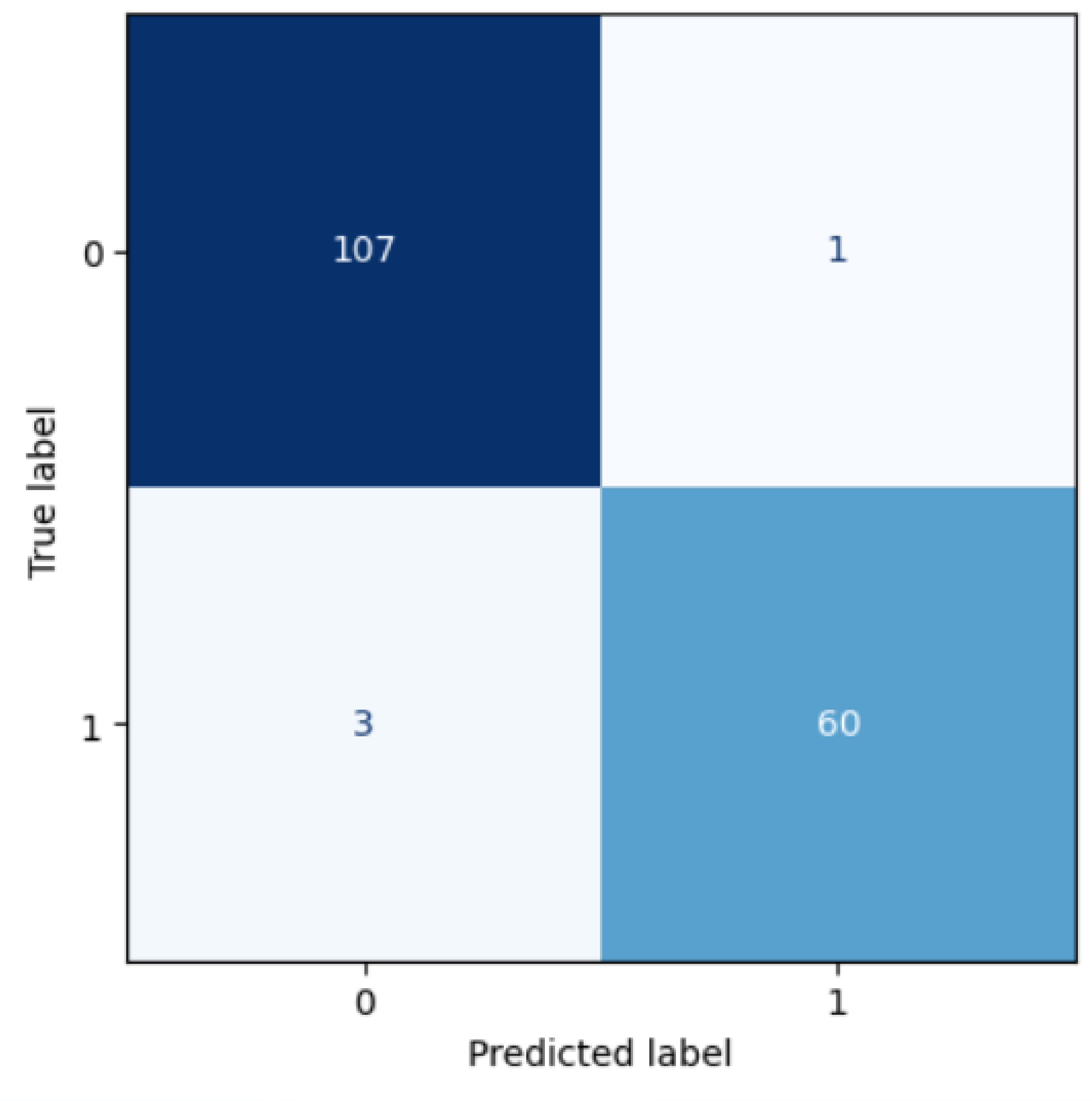
- Scaling (standardization or normalization) is required when we use any machine learning algorithm that require **gradient calculation.**

TRAINING & TESTING MODELS

TRAINING & TESTING MODELS

Logistic regression

```
from sklearn.linear_model import LogisticRegression

# training the model
logModel = LogisticRegression()
logModel.fit(X_train, y_train)
pred = logModel.predict(X_test)
```



0.9766 accuracy

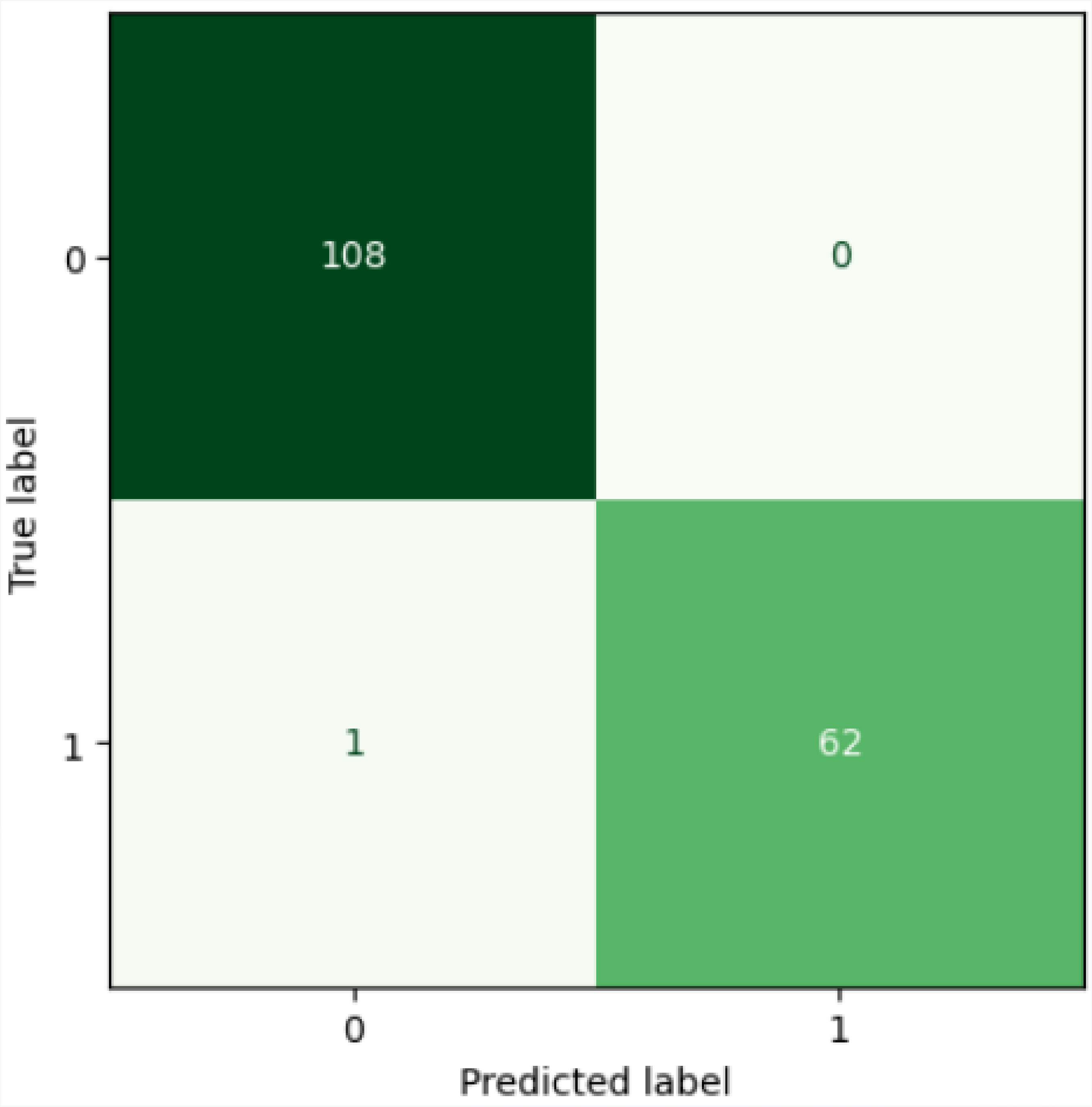
TRAINING & TESTING MODELS

Neural Network

```
# importing tensorflow and Keras
import tensorflow as tf
tf.random.set_seed(3)
from tensorflow import keras
```

```
# setting up the layers of Neural Network

NN_model = keras.Sequential([
                            keras.layers.Flatten(input_shape=(30,)),
                            keras.layers.Dense(60, activation='relu'),
                            keras.layers.Dense(2, activation='sigmoid')
])
```

```
# compiling the Neural Network

NN_model.compile(optimizer='adam',
                loss='sparse_categorical_crossentropy',
                metrics=['accuracy'])
```



0.9942 accuracy

TRAINING & TESTING MODELS

Gradient Boosting Classifier

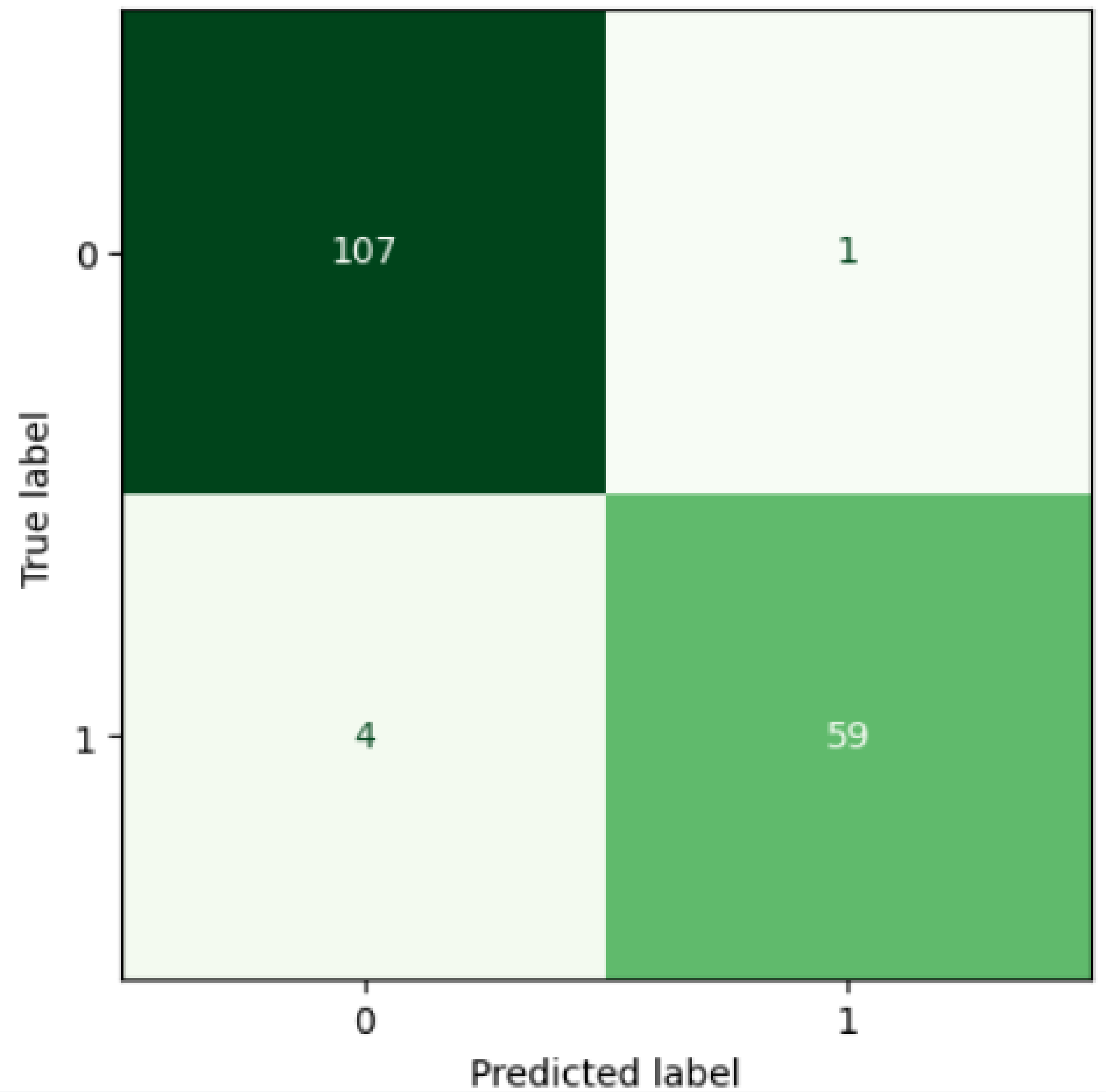
```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV

gbc = GradientBoostingClassifier()

parameters = {
    'loss': ['deviance', 'exponential'],
    'learning_rate': [0.001, 0.1, 1, 10],
    'n_estimators': [100, 150, 180, 200]
}

grid_search_gbc = GridSearchCV(gbc, parameters, cv = 5, n_jobs = -1, verbose = 1)
grid_search_gbc.fit(X_train, y_train)
```

```
# best parameters
grid_search_gbc.best_params_
```

```
{'learning_rate': 1, 'loss': 'exponential', 'n_estimators': 100}
```

```
gbc = GradientBoostingClassifier(learning_rate = 0.1, loss = 'exponential', n_estimators = 180)
gbc.fit(X_train, y_train)
```

```
GradientBoostingClassifier
GradientBoostingClassifier(loss='exponential', n_estimators=180)
```



0.97 accuracy

TRAINING & TESTING MODELS

Stochastic Gradient Boosting (SGB)

```
sgbc = GradientBoostingClassifier(max_depth=4, subsample=0.9, max_features=0.75, n_estimators=200, random_state=0)
sgbc.fit(X_train, y_train)
```

K Neighbors Classifier (KNN)

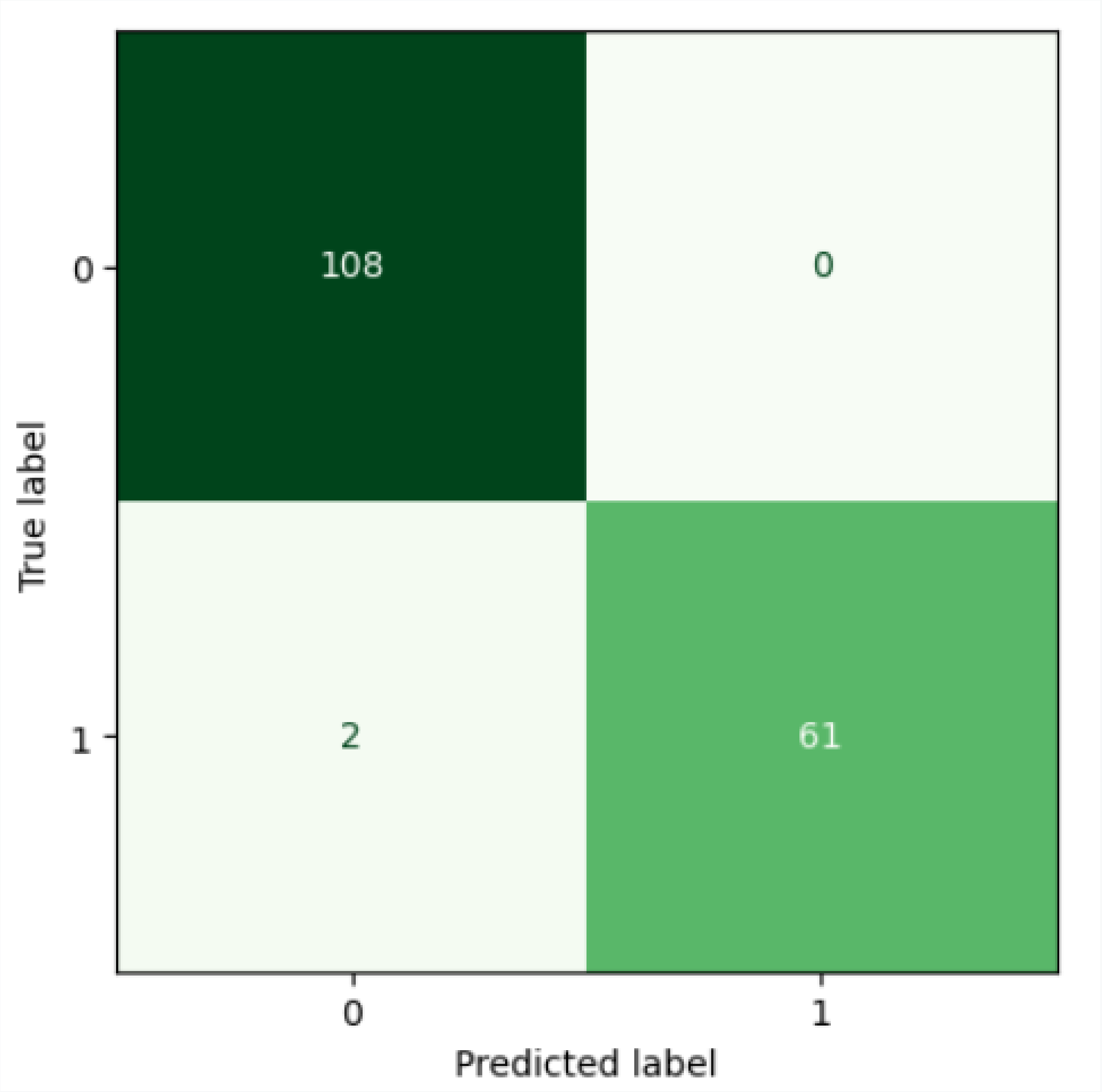
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
```

the same accuracy & and confusion Matrix as GBC

TRAINING & TESTING MODELS

Support Vector Classifier (SVC)

```
:': 20, 'gamma': 0.001}
```

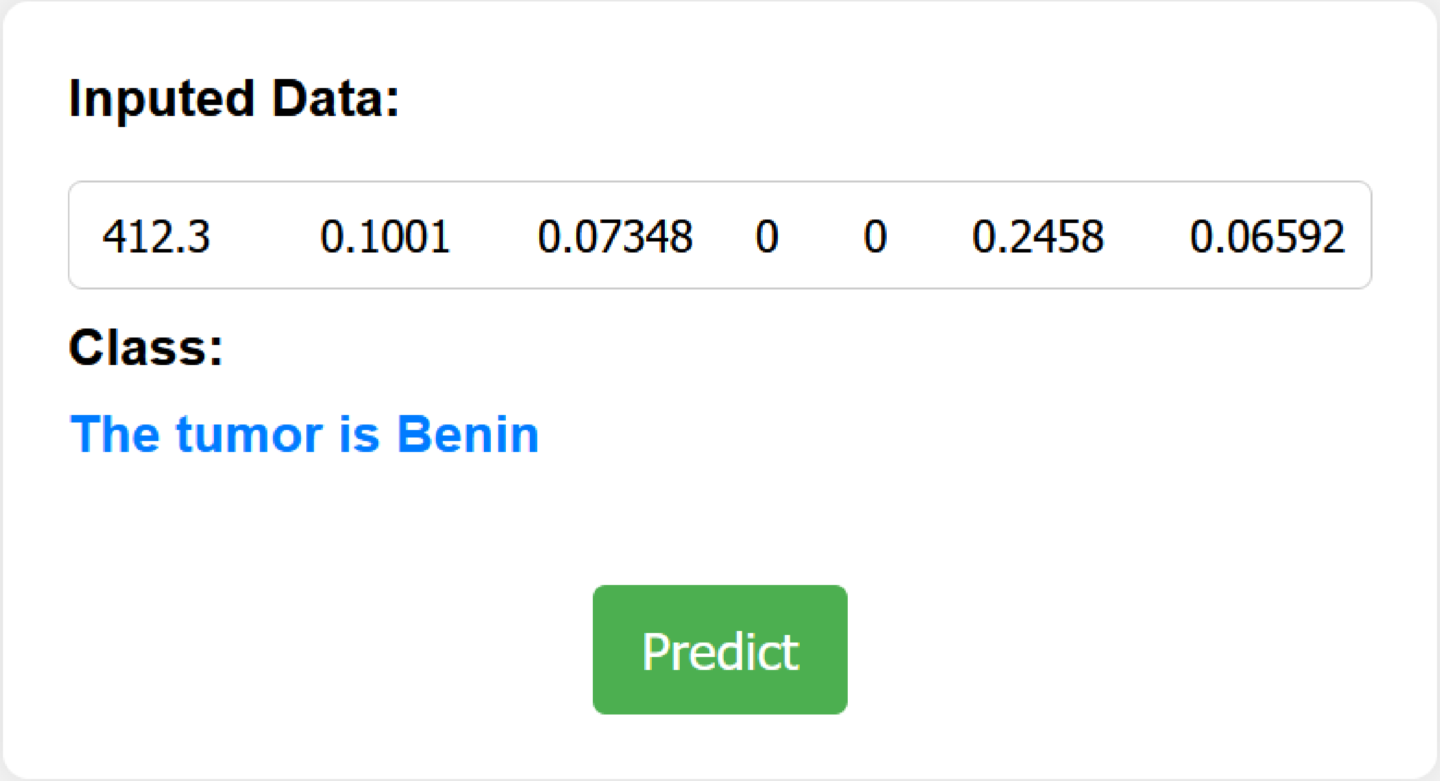


0.988 accuracy

Neural Network

0.9942 accuracy

Model Deployment using Flask FW



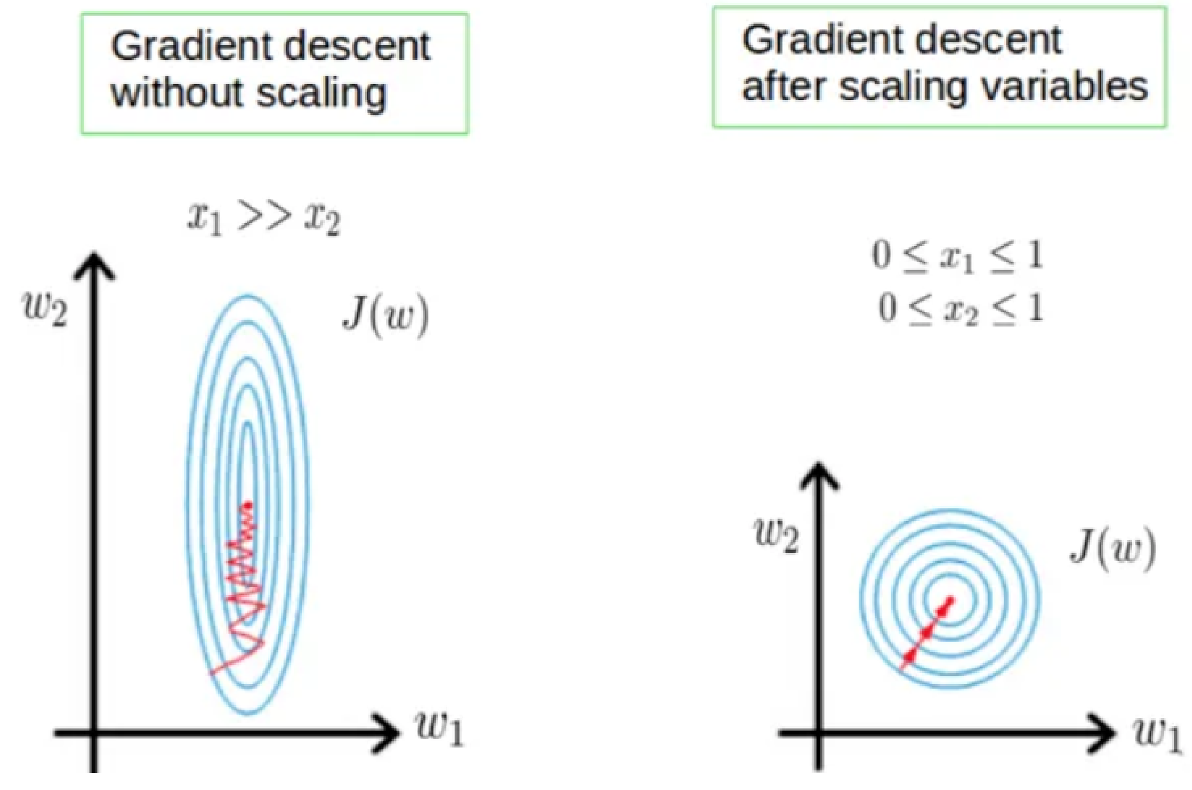
CHALLENGES

CONCLUSION

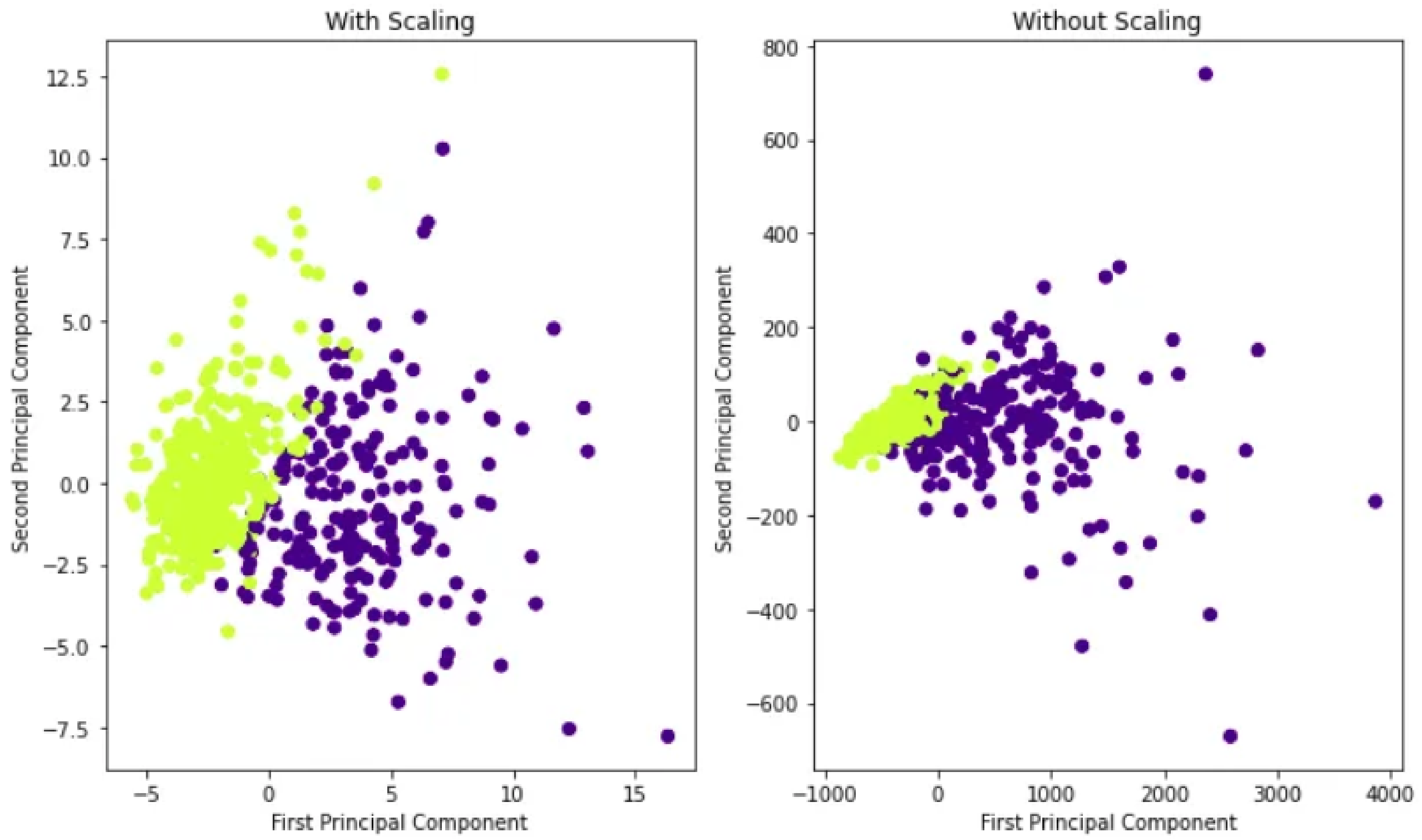
FUTURE WORK !

CNN implementation from scratch

1. Mean Centering: Subtracting the mean from each feature ensures that the mean of the feature becomes zero. This is important because it removes any bias from the features. When features have different means, they can bias the learning algorithm, especially in algorithms that are sensitive to feature scales, like gradient descent-based algorithms.
2. Variance Scaling: Scaling the features to unit variance means that the variance of each feature becomes 1. This is important because it gives equal importance to all features, preventing features with larger variances from dominating the learning process. It also helps algorithms converge faster, especially for algorithms that are sensitive to feature scales.


Gradient descent without scaling
Gradient descent after scaling variables
$x_1 >> x_2$
$0 \le x_1 \le 1$
$0 \le x_2 \le 1$
w_2
w_1
$J(w)$

Reference


With Scaling
Without Scaling
Second Principal Component
First Principal Component