

**A Comprehensive Analysis of Breast Cancer Prediction Using Neural Networks**

STW7088CEM Artificial Neural Networks

Name: Umanga Niroula

ID: 14825284

Submitted To: Subodh Acharya

Table of Contents

[1 Introduction 4](#_Toc173624731)

2 [Literature Review 5](#_Toc173624732)

3 [Problem Statement and Task Performed 8](#_Toc173624733)

[3.1Problem Definition 8](#_Toc173624734)

[3.2Tasks Performed 8](#_Toc173624735)

[3.2.1 Exploratory Data Analysis (EDA) 8](#_Toc173624736)

[3.2.2 Data Preprocessing 16](#_Toc173624737)

[3.2.3 KMeans Clustering and Dimensionality Reduction 21](#_Toc173624738)

[3.2.4 Classification 22](#_Toc173624739)

[4 Discussion of Findings 28](#_Toc173624740)

[5 Conclusion 29](#_Toc173624741)

[6 References 30](#_Toc173624742)

[7 Appendices 32](#_Toc173624743)

**List of Figures**

[Figure 1: Loading Data into a Data Frame. 9](#_Toc173624898)

[Figure 2: Number of Rows and Columns in the Dataset. 9](#_Toc173624899)

[Figure 3: Top and Bottom Five Rows of the Dataset. 10](#_Toc173624900)

[Figure 4: Information on the Columns of the Dataset. 11](#_Toc173624901)

[Figure 5: Count of Null Value for each of the Columns. 12](#_Toc173624902)

[Figure 6: Counting for Duplicate Records in the Dataset. 13](#_Toc173624903)

[Figure 7: Box Plot and Histogram Plot for Radius Mean Column. 13](#_Toc173624904)

[Figure 8: Box Plot and Histogram Plot for Texture Mean Column. 14](#_Toc173624905)

[Figure 9: Box Plot and Histogram Plot for Perimeter Mean Column. 14](#_Toc173624906)

[Figure 10: Box Plot and Histogram Plot for Area Mean Column. 15](#_Toc173624907)

[Figure 11: Box Plot and Histogram Plot for Concave Points Worst Column. 16](#_Toc173624908)

[Figure 12: Heatmap of Feature Correlations. 17](#_Toc173624909)

[Figure 13: Code Snippet to Drop Unwanted Columns from the Dataset. 17](#_Toc173624910)

[Figure 14: Information on the Columns after Dealing with Outliers. 18](#_Toc173624911)

[Figure 15: Box Plot and Histogram Plot for Radius Mean Column after Dealing with Outliers. 19](#_Toc173624912)

[Figure 16: Box Plot and Histogram Plot for Texture Mean Column after Dealing with Outliers. 19](#_Toc173624913)

[Figure 17: Box Plot and Histogram Plot for Area Mean Column after Dealing with Outliers. 20](#_Toc173624914)

[Figure 18: Splitting Dataset into Feature and Target. 20](#_Toc173624915)

[Figure 19: Dataset which Contains all of the Feature Columns. 21](#_Toc173624916)

[Figure 20: Dataset which Contains the target Columns in Numeric Format. 21](#_Toc173624917)

[Figure 21: Normalizing the Feature Columns. 22](#_Toc173624918)

[Figure 22: Principal Component Analysis of Clusters in 2D. 23](#_Toc173624919)

[Figure 23: Size of Training Data for Feature and Target Dataset. 23](#_Toc173624920)

[Figure 24: Size of Testing Data for Feature and Target Dataset. 24](#_Toc173624921)

[Figure 25: Building the Neural Network Model. 24](#_Toc173624922)

[Figure 26: Summary of the Neural Network Model. 25](#_Toc173624923)

[Figure 27: Training the Model. 26](#_Toc173624924)

[Figure 28: Plotting the Accuracy and Loss of the Model for Training and Validation Dataset. 27](#_Toc173624925)

[Figure 29: Confusion Matrix for the Model. 28](#_Toc173624926)

# Introduction

Breast cancer in this generation is one of the most alerted health concerns amongst women globally hence, early detection is crucial to adequate prevention and treatment. The earlier breast cancer is found, the better chace of survival as the preventive measures can be taken if the breast cancer is found before it starting to cause major damage to patients. Machine learning algorithms have come wide ways and now, they enable us to make even better predictions on earlier stages of breast cancer which have ultimately saved numerous lives. The American Cancer Association (2021) states that through this strong algorithm, they can detect early breast cancer and intervene before symptoms appear. As far as the medical field is concerned, machine-learning and neural-network-based algorithms have been making a major difference as it has been able to detect various disease accuracy and faster than any human being. These tools minimize the burden on healthcare providers and help patients to start preventive measures early.

This report presents different machine learning and neural network methods to identify with high accuracy whether breast cancer is benign or malignant. Benign is used in non-cancerous conditions and malignant refers to cancerous condition. Classification uses the Wisconsin Breast Cancer Dataset for clustering, available from UCI Machine Learning (Dua & Graff, 2019). This report will use the dataset to answer how these technologies can be leveraged in practice with computer algorithms and simulate approaches that are more broadly applicable to classify a diagnosis of medical conditions using data obtained therein. In this report, we demonstrate the power of machine learning and neural networks for breast cancer detection and classification using analysis and experimentation.

# Literature Review

There have been great advancements made in consideration of utilizing machine learning techniques to solve medical-related problems which have helped in medical research and clinical practices over many years. Sidey-Gibbons & Sidey-Gibbons 2019 is not the only research group following this approach of using machine learning in medical contexts, yet attention on both a conceptual understanding as well as implementing it. In this larger framework is the work of Treu et al. which deals with building predictive algorithms for cancer diagnosis based on a dataset containing samples of both breast cancer and non-cancer records. The authors developed three models, namely regularized General Linear Model Regression, SVM, and Artificial Neural Networks which are single-layered. They train-and-tested-train-thusly their models by doing a cut of the data through an evaluation & validation sample.

Tanaka & Aranha (2019) also do similar work of experimenting with Generative Adversarial Networks to generate data in their research paper which can give us insight, into how it is useful for training data learning. By replacing the example generation mechanisms with a novel, unsupervised method of GANs (if appropriate for the problem at hand), new techniques like SMOTE or ADASYN that over-sample minority classes and preserve privacy. Their work tested the strength of a GAN by evaluating benchmark datasets with different network architectures. Most noteworthy of all, the recall measure was used to assess how well a model not only performed but was able to detect positive instances equally. Two key rationales for generating synthetic datasets are cited in this article by the authors. Instead, it deals with imbalanced datasets as found in credit card fraud detection or medical diagnosis where minority classes might be rare. Such an approach lets the classifier learn minority classes more effectively and avoid overfitting since it comes up with new augmentation data to be generated out of such underrepresented instances. That means that some of your data might be becoming non-compliant with this new regulation.

Chae & Wilke 2019 also investigate the subset selection of step sizes in the mini-batch sub-sampling (MBSS) for neural network training. You can run the code in two modes: - static mode (where you update mini-batch only after changing the direction of the search). In this study, the second-order quadratic line search corrections are used to ascertain how well that function and derivative information help in constructing these approximations for dynamic MBSS loss functions. Experimented with many neural network tasks, and found that carefully selecting enforced information leads to up-to several orders of magnitude improvements in prediction accuracy in step sizes. These results provide critical insights into a trade-off between bias and variance in MBSS as well as plausible settings of step size dependency that are necessary for defining optimal parameter selection, which is indispensable to the margin (i.e. fastest learning) NNs.

Van Looveren and Klaise (2019) similarly propose a fast, model-agnostic approach to finding interpretable counterfactual explanations of classifier predictions by using class prototypes. The approach is built on top of either a prototype from an encoder or class-specific k-d trees to speed up the process. The approach was tested on image (MNIST) and tabular (Breast Cancer Wisconsin Diagnostic). Counterfactuals: Desire for Interpretability Quyen Luu, Hai Dang, and Shuo Wang tackle the problem of balancing sparsity vs Training desired characteristic notes with interpretability in counterfactual instances. They point out that sparse perturbations can be closer to the underlying manifold when looking at data as a whole, but may not be aligned with one of the two subset classes for which you wish your counterfactual example. The proposed method addresses this issue by including class prototypes in the loss function, avoiding computational bottlenecks related to numeric gradient computations for black-box models. This work is a major step forward in being able to produce human-interpretable counterfactual explanations and provides an interpretable framework that can be used with different data sets and model types.

# Problem Statement and Task Performed

## Problem Definition

In the current context many of the women have fallen victim to breast cancer, since breast is a very private part many of the patient suffering from such problems never show up early for checkup which increases the risk of them causing serious damage to themselves in the near future. Likewise, for those that do go to check might be miss judged as their breast cancer has not developed to a significant level. For example, is a women might have breast cancer but in a very small proportionate and if the doctors make a mis judgment as labels her as non-cancerous then she might not start any preventive measures risking her life. In such cases a need of accurately identifying cancer is much in need.

## Tasks Performed

### Exploratory Data Analysis (EDA)

1. Basic Exploration:

We create a data frame (df) where the data’s reference is stored as shown by the following figure 1.



Figure : Loading Data into a Data Frame.

Now to begin the basic data exploration, the shape of the data is checked as shown by the following figure 2. The figure shows that there are 569 rows and 33 columns.



Figure : Number of Rows and Columns in the Dataset.

Now the head and tail of the dataset are shown which provides the top and bottom 5 rows of the dataset as demonstrated by the following figure 3.

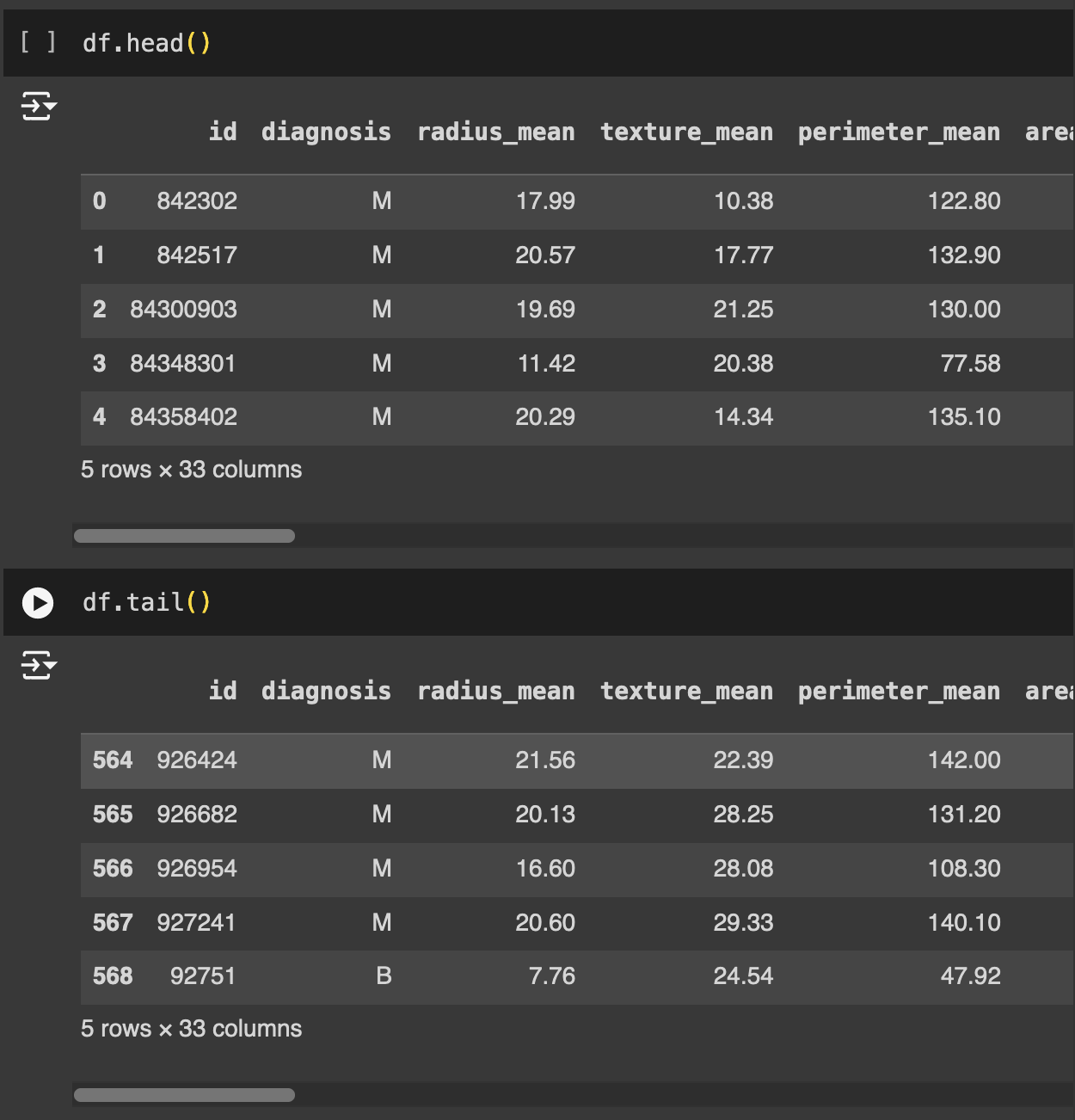


Figure : Top and Bottom Five Rows of the Dataset.

Since, there are all together 33 columns in the dataset as shown by figure 2. With the help of this we can separate which are the feature columns and which is the target one. Here, the “diagnosis” column is the target column. This column consists of two values, ‘M’ which indicates cancer or ‘B’ which states non-cancer. Further rest of the 32 columns help to identify weather it is of M or B category

Now, all the information on the each of the 33 rows are provided in the following figure 4. Out of all the 33 columns the diagnosis column is of the data type “object”, which is a categorical value. Likewise, rest of all are numerical datatype of either integer or float. Through this analysis we can also understand which is the target columns and which are the feature column.

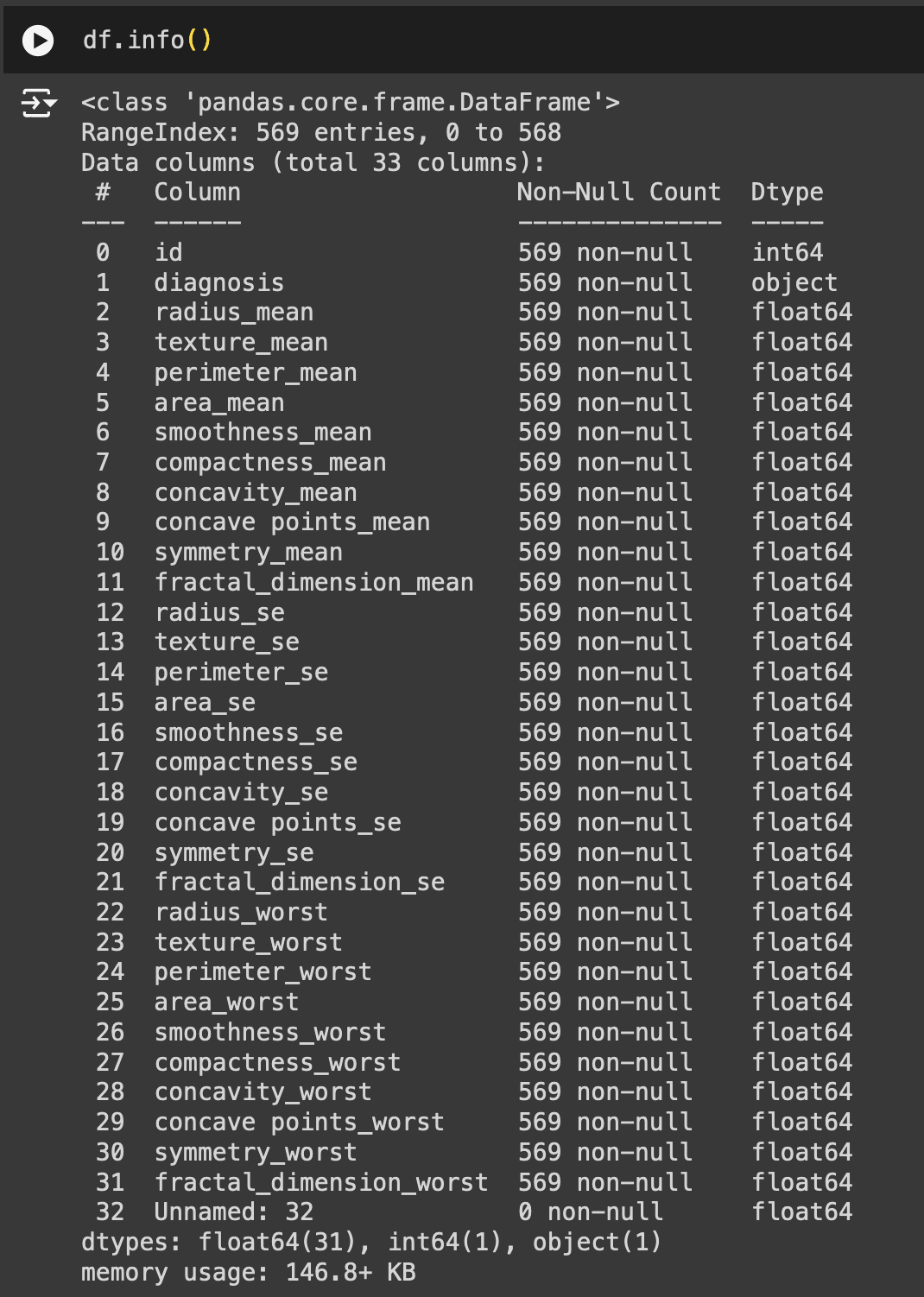


Figure : Information on the Columns of the Dataset.

Further, the figure 4 shows the null-count of each of the columns, which also shows that there is no missing value in any of the columns. To see the count of null value properly following figure 5 shows the details.



Figure : Count of Null Value for each of the Columns.

The figure 5 shows that the ‘Unnamed :32’ column has a total of 569 null value which shows that this column is completely useless and can be dropped during the time of data cleaning.

Also, to ensure there are no duplicate records, the dataset must be checked. The following figure shows that there are no duplicate records found in the dataset.

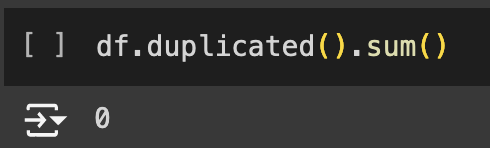


Figure : Counting for Duplicate Records in the Dataset.

With the checking of the duplicate data, the basic exploration of the data is completed. Now more complex exploration will be conducted where outliers in the dataset, also the confusion matrix of the data will also be generated to see the correlation between each of the columns such that to check the dependency of the target columns “diagnosis” with the rest of the features (Chandola et al., 2009; Provost & Fawcett, 2013). Further, correlation helps us in identifying which columns are dependent in the target column. The dependency shows that which column is most important in correctly predicting the target column(Guyon & Elisseeff, 2003).

1. Outliers Identification:

Now to identify the outlies, we take the help of box plot and histogram. For all of the 31 columns, outliers were visualized using box plot and bar graph. Following figure 7 shows the visualization for outliers.

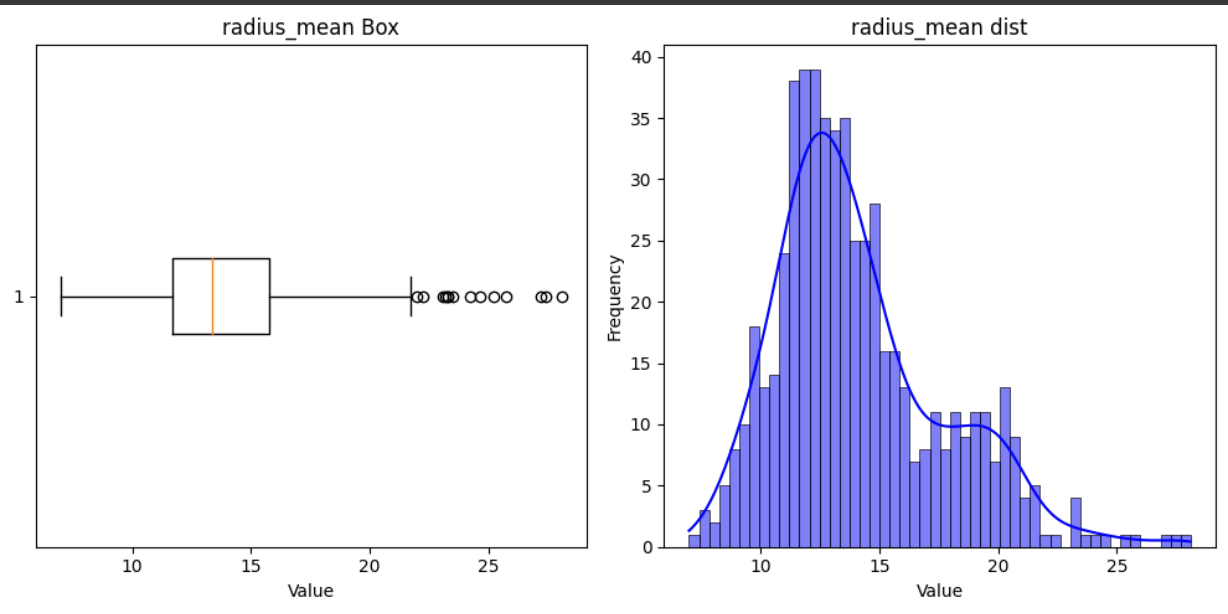


Figure : Box Plot and Histogram Plot for Radius Mean Column.

Likewise, the same graph for texture means, perimeter mean, and area mean which are few of the columns form the 31-column list are shown in the following figure 8, 9, and 10 respectively.

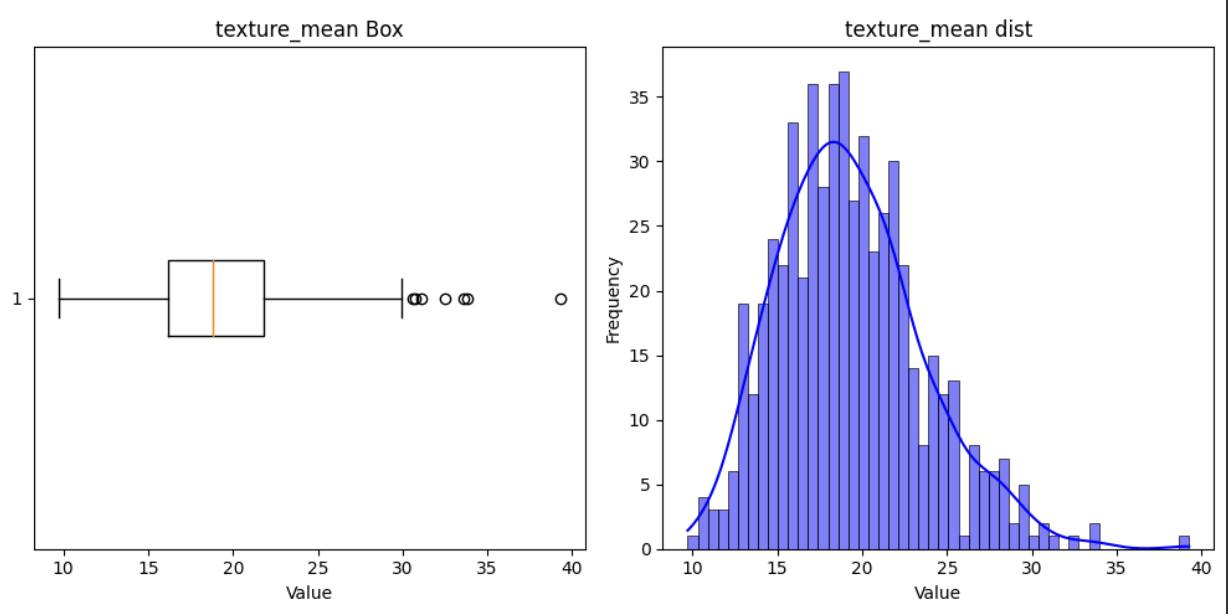


Figure : Box Plot and Histogram Plot for Texture Mean Column.

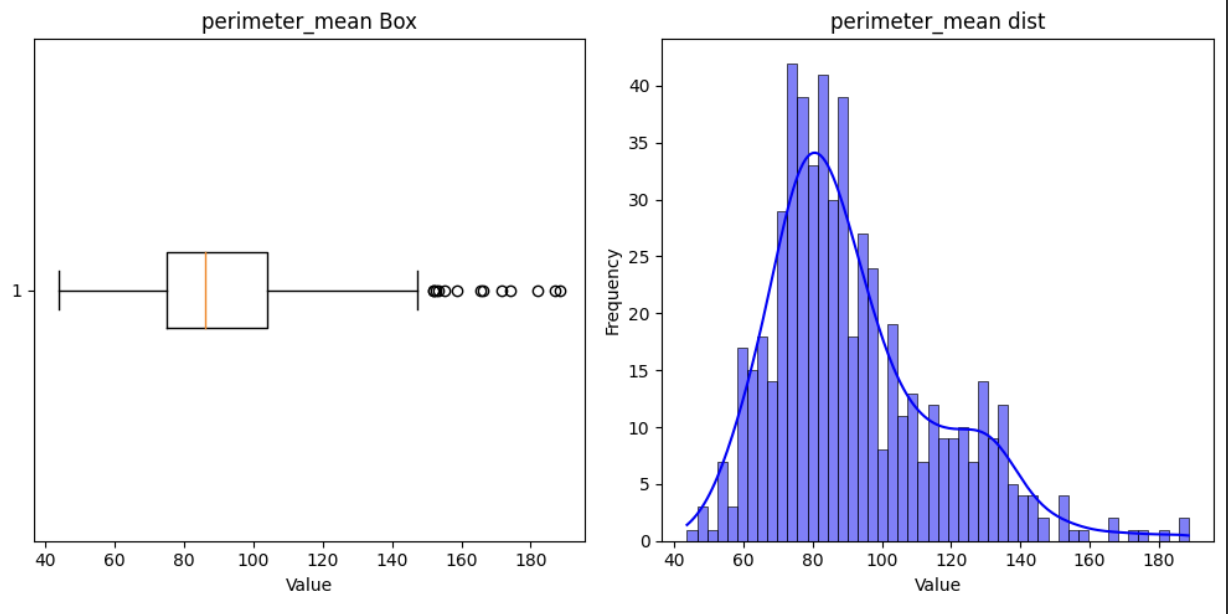


Figure : Box Plot and Histogram Plot for Perimeter Mean Column.

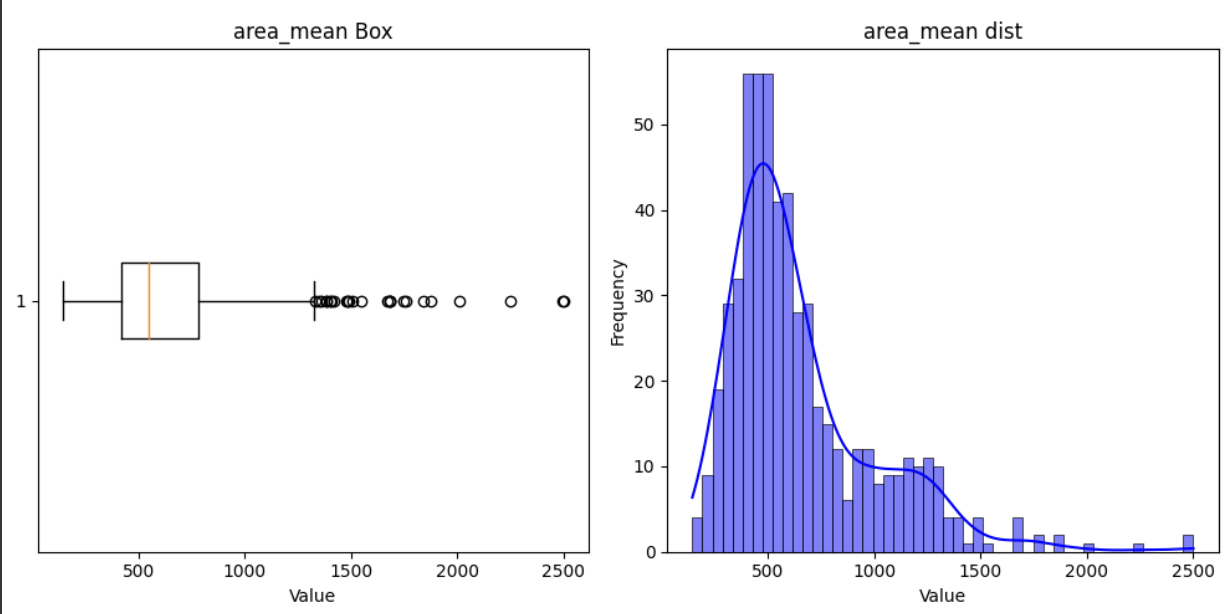


Figure : Box Plot and Histogram Plot for Area Mean Column.

Further, the reason to pick 31 columns out of 33 is because, one of the columns is a target column meaning its data type is categorical. Furthermore, outliers can only be identified for numerical value not categorical. Likewise, other columns in the dataset contains id, where there are no outliers. So, rest of the 31 columns which consists float data type value were taken for showing the visualization. Moreover, for identifying outliers with the help of box plot is very easy. They are represented by dots outside of the wishker as shown in figure 7, 8, 9 ,10. All of the dots shown in the four points represent outliers as they are present outside of the whiskers. Similar kind of points can be seen for other columns in the dataset. Likewise, the histogram on the right side can also be used to identify outliers as certain tails of the distribution are tall indicating that there are outliers within the column.

Now the following figure 11 shows the graph for concave points worst column where there is no point outside of the whiskers. Hence, this suggests that there are no outliers in this column. So out of 31 columns 30 columns consist of outliers.

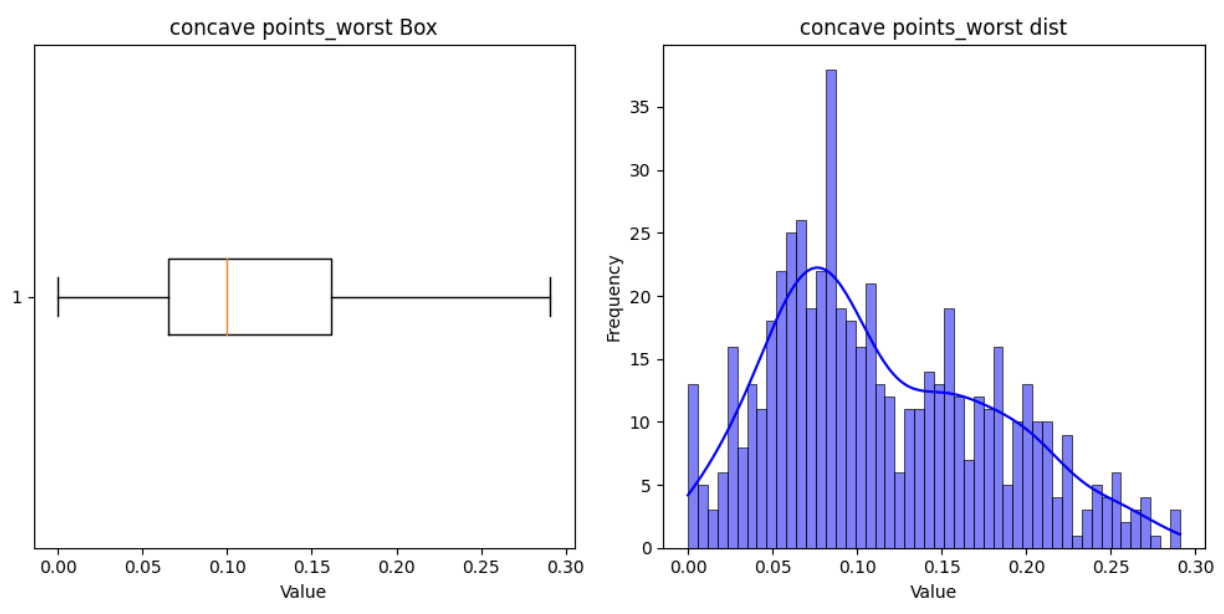


Figure : Box Plot and Histogram Plot for Concave Points Worst Column.

1. Correlation Matrix:

Now to end the exploration phase, a heat map has been generated for all the 32 columns to check the correlation of each of the columns with one another. This is specially used to check the dependencies of 32 columns with the target column. The following figure 12 shows the heat map.

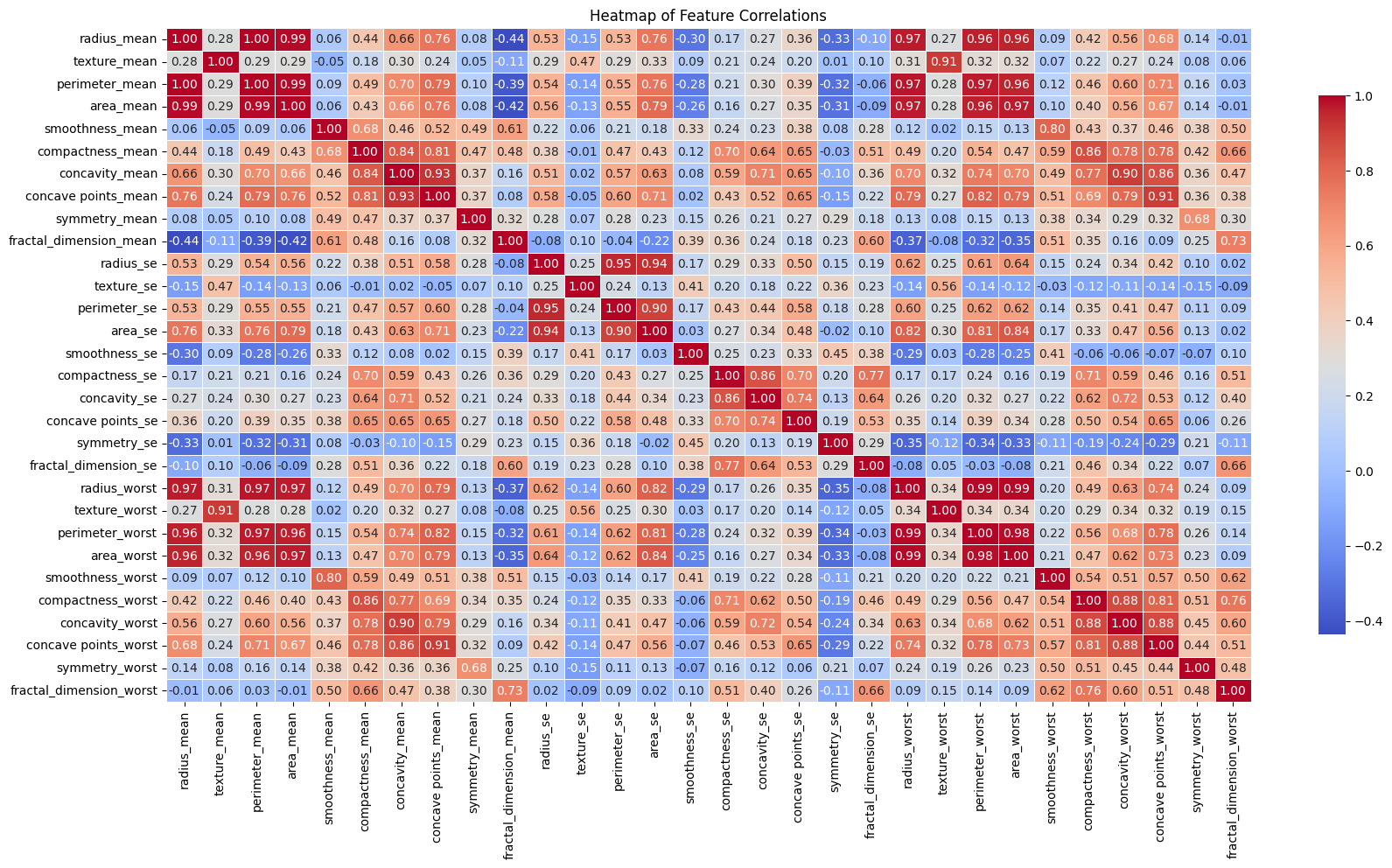


Figure : Heatmap of Feature Correlations.

The heatmap shows no such weak or strong relation between any of the features, suggesting that each of the columns contribute in certain ways to positively or negatively influence thr target column.

### Data Preprocessing

1. Dropping Unwanted Columns:

In the exploration phase, we saw that the ‘unnamed: 32’ contains only null values which shows that it has no importance in helping to predict the target value of ‘diagnosis’ columns, which is why the first thing in the data preprocessing is removing the unwanted columns to make the data more accurate. Further, the following figure 13 shows the code snippet used to drop the unwanted column which contained only null value.

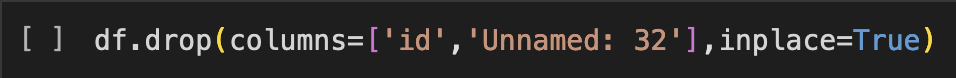


Figure : Code Snippet to Drop Unwanted Columns from the Dataset.

1. Handling Outliers:

Now since the outliers were identified using the interquartile ranger method, those records that consisted of outliers were dripped form the data frame. This method ensures that only reasonable records or values are present inside the data frame which ensure more accuracy during the time of building model. Further, after dropping the outliers form the data frame the following figures shows the information about the outlier free data frame. As shown by the figure 14 there are only 398 records which was initially 569 as seen in figure 4. Further, the comparison between figure 14 and 4 shows that many of the data were dropped due to them identifying as outliers.

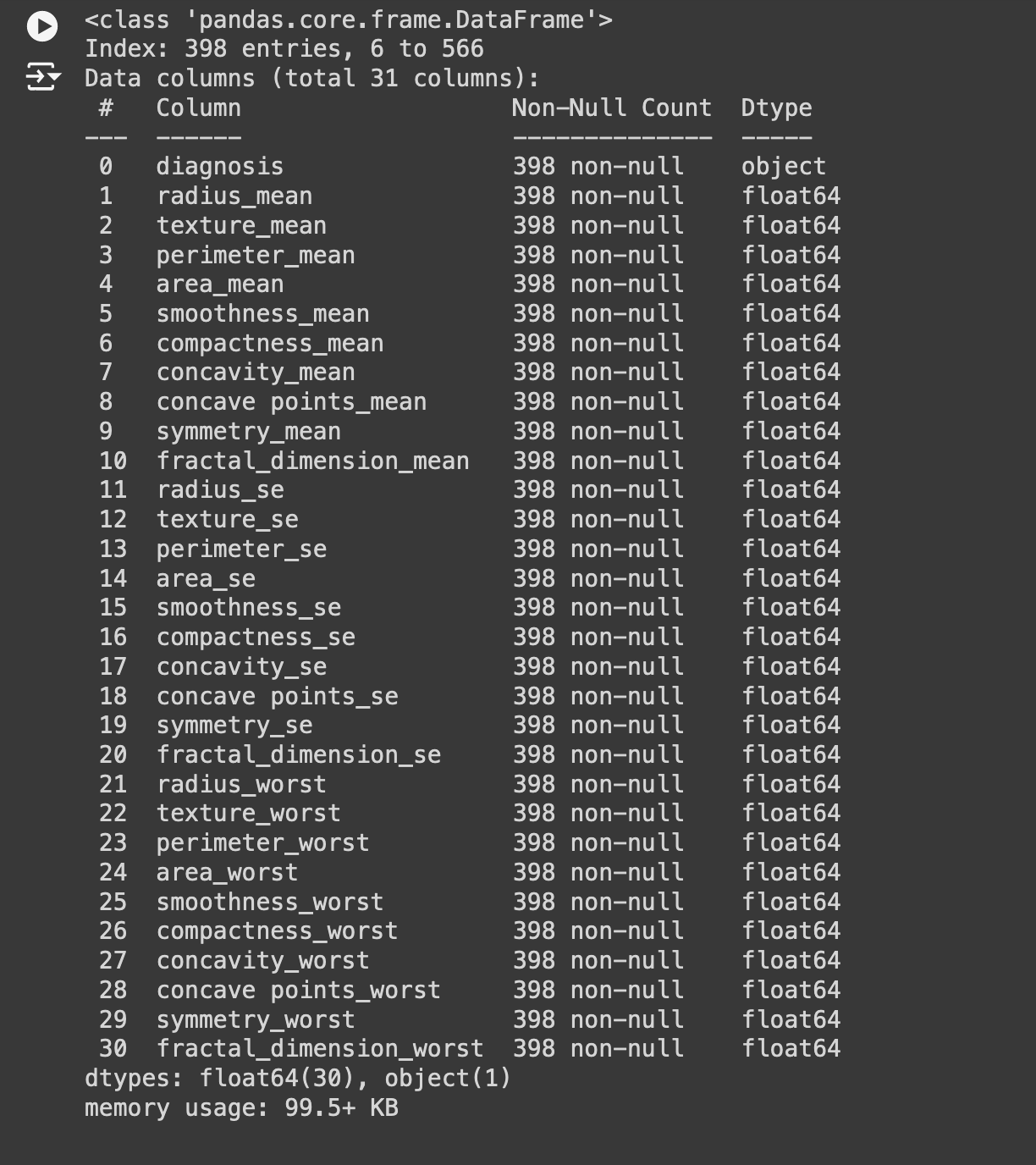


Figure : Information on the Columns after Dealing with Outliers.

Now after the outliers were dropped form the dataset, the visualization were again generated to see the difference after dealing with the outliers. The following figure 15, 16, and 17 shows the visualization for Radius Mean, Texture Mean, Perimeter Mean, and Area Mean respectively.

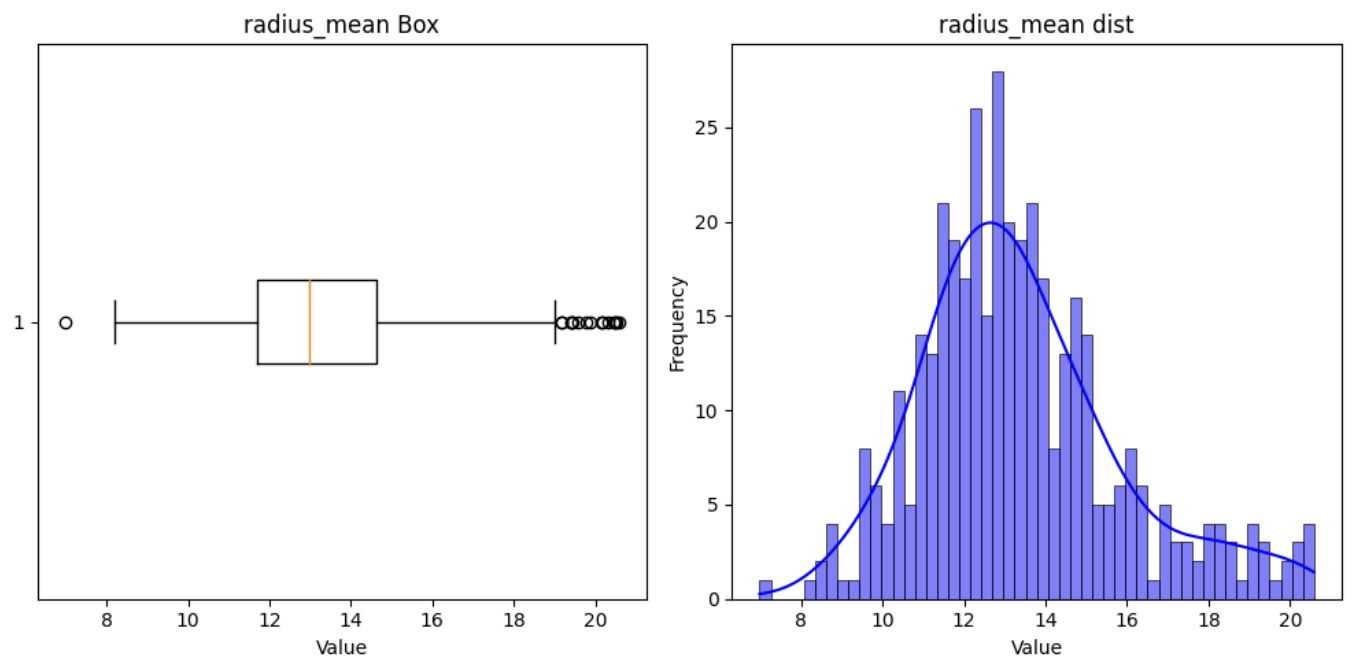


Figure : Box Plot and Histogram Plot for Radius Mean Column after Dealing with Outliers.

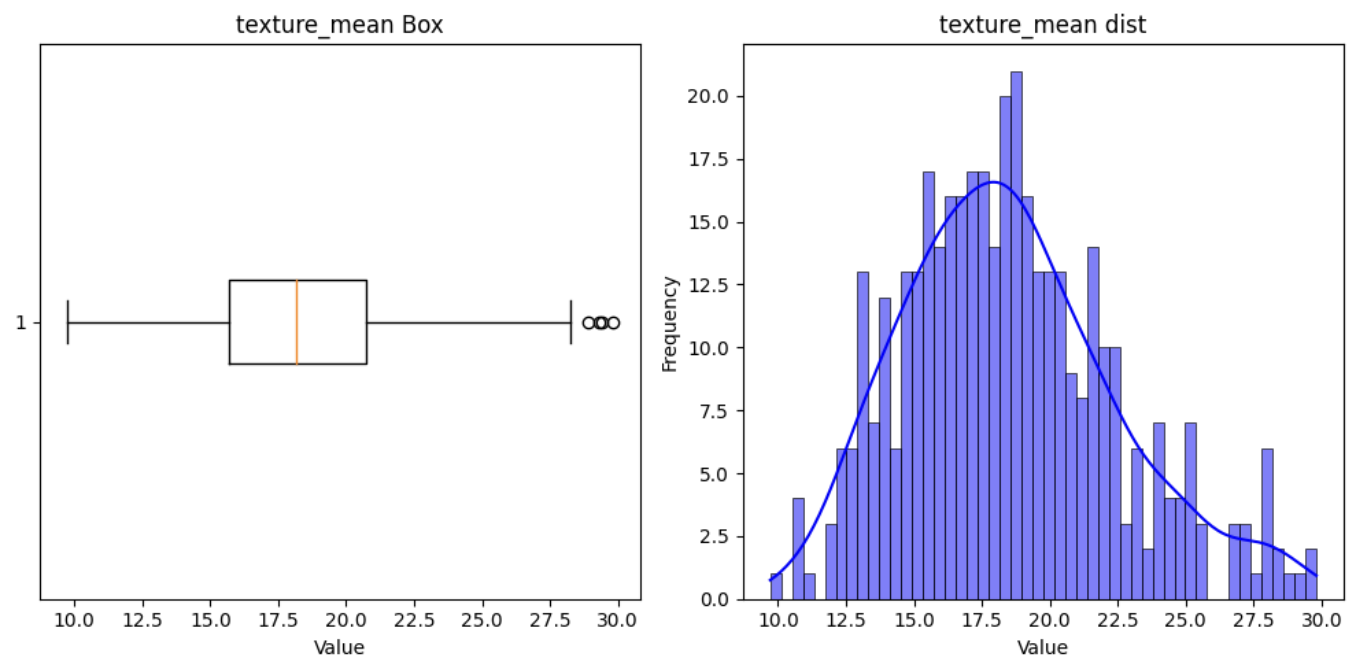


Figure : Box Plot and Histogram Plot for Texture Mean Column after Dealing with Outliers.

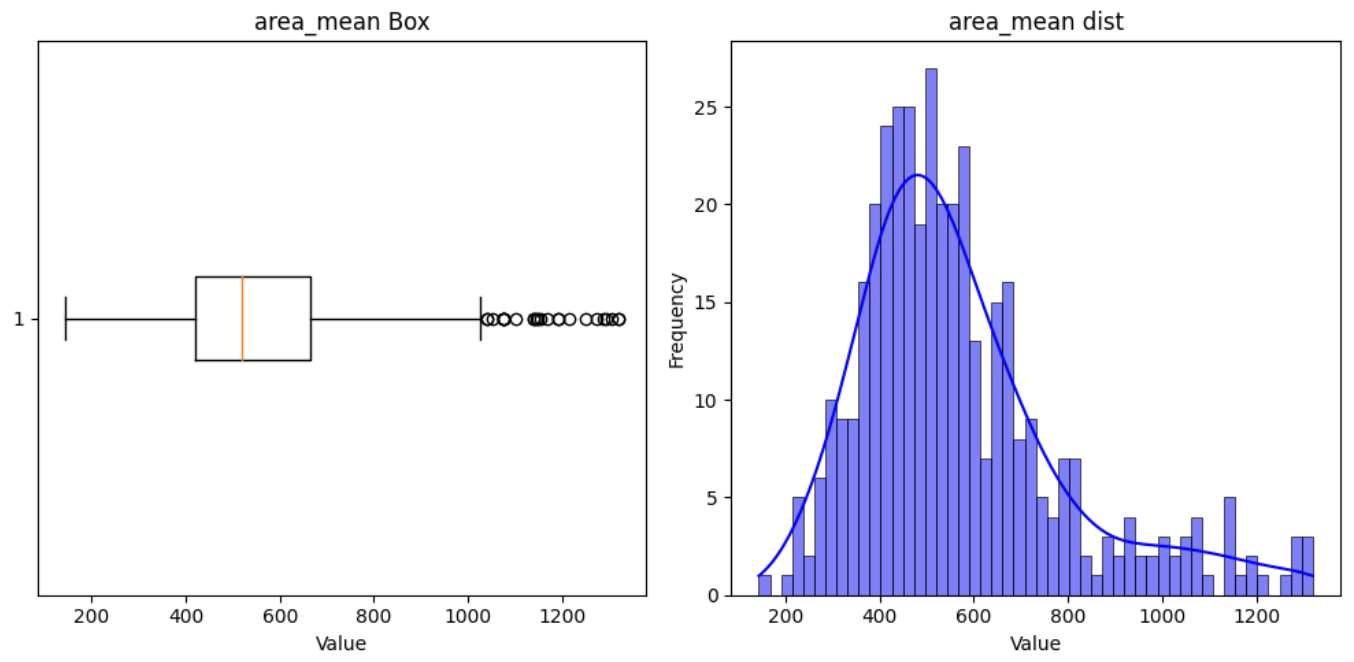


Figure : Box Plot and Histogram Plot for Area Mean Column after Dealing with Outliers.

Now comparing figure 7 and 15 we can see that the upper limit of the outliers is comparatively less after dealing with the outliers. Similarly, the difference can be visually seen while comparing 8-16 and 9-17. This way it is certain that now the dataset doesn’t contain any outliers.

1. Features Target Splitting:

After cleaning the dataset, now we split the dataset into two parts; one features which is represented by ‘X’ and target which is represented by ‘y’, meaning all of the 31 columns are present inside X and y contains just the diagnosis column. The following figure 18 shows the division of dataset into two portions.

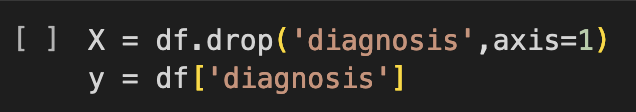


Figure : Splitting Dataset into Feature and Target.

This division is done as it is very necessary step before training a machine learning model.. The following figure 19 shows all the dataset which is contained inside X.

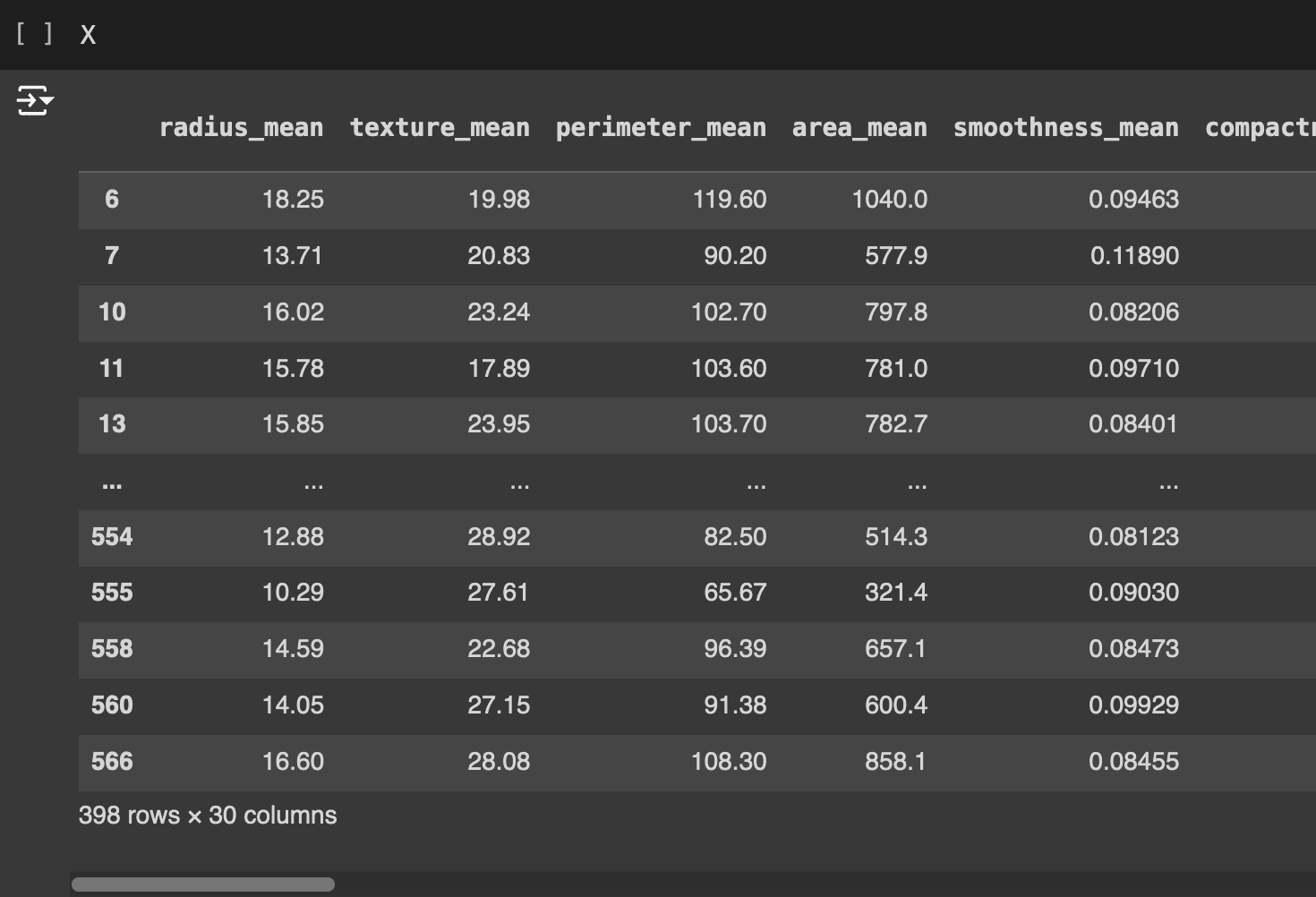


Figure : Dataset which Contains all of the Feature Columns.

1. Encoding Categorical Variables:

Likewise, the figure 20 shows what ‘y’ contains which is the target value. Here the target variable is converted from categorical value to numerical value where 1 represent cancer and 0 represents non-cancer.

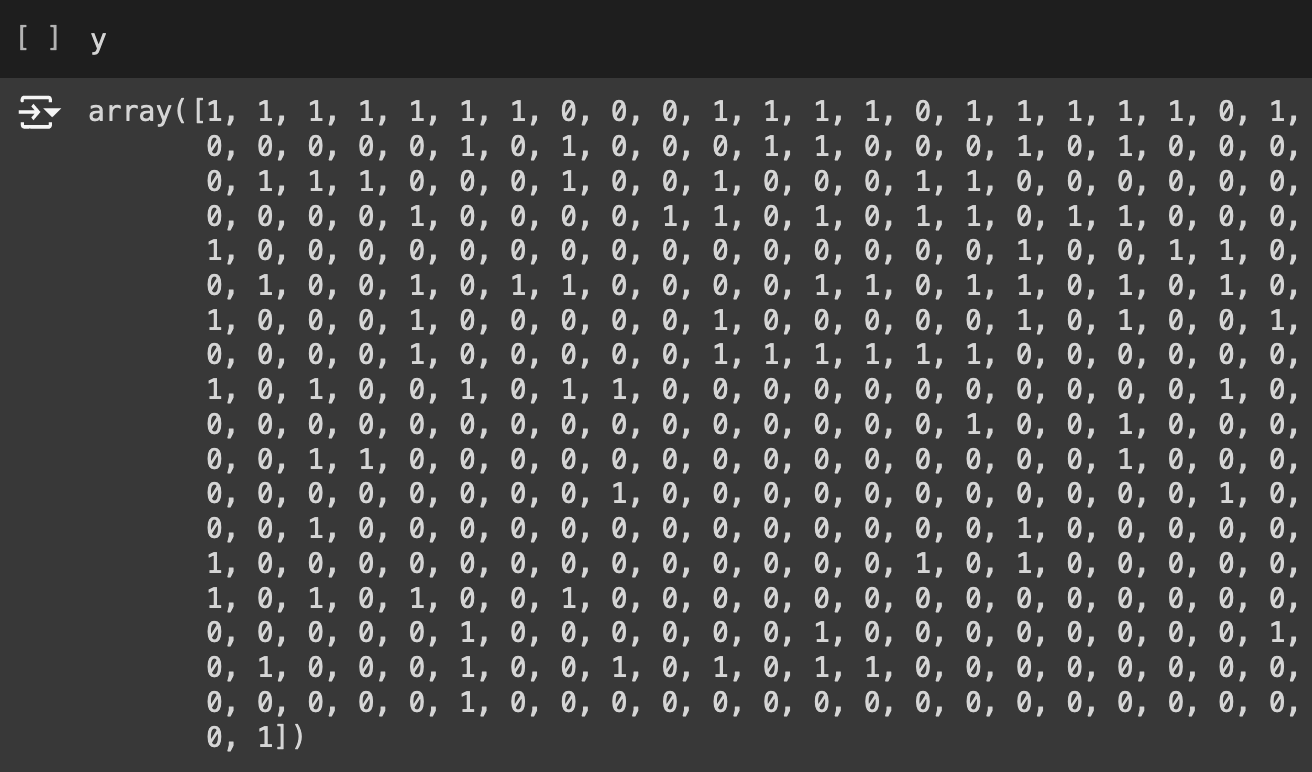


Figure : Dataset which Contains the target Columns in Numeric Format.

1. Scaling Features:

Now since, the target and feature columns are separated, without the target column there are big difference between the numerical data as shown in figure 19. If these larger values are not scaled then larger values might dominate the smaller ones. The following figure 21 shows that all of the feature values have been normalized using MinMaxScaler.

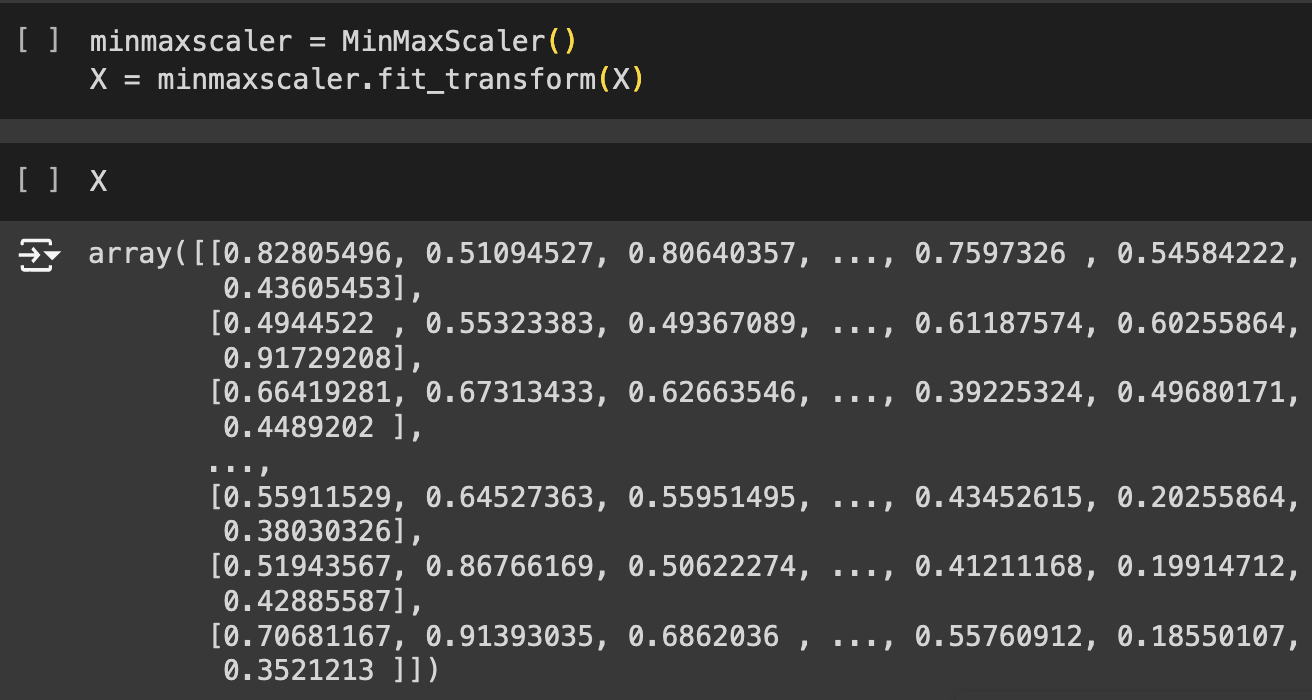


Figure : Normalizing the Feature Columns.

### KMeans Clustering and Dimensionality Reduction

First we keep the feature dataset into a KMeans clustering algorithms followed by Principal Component Analysis (PCA) to reduce dimensionality along with visualizing the data into two dimensions. Further, this is done to understand the distribution and structure of the data which is also shown in terms of scatter plot by the following figure 22.

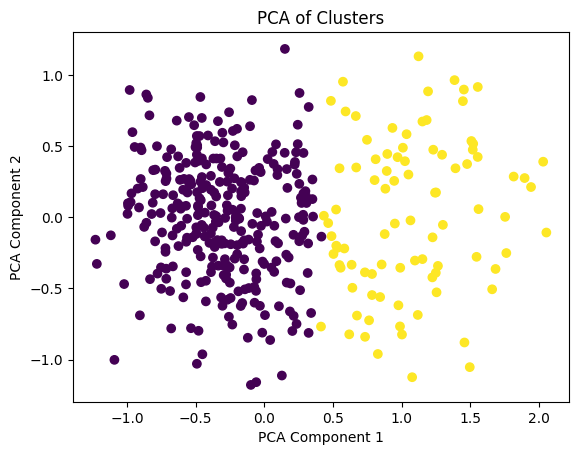


Figure : Principal Component Analysis of Clusters in 2D.

### Classification

1. Train and Test Split:

The first step before building a model is to further divide the feature and target dataset into train and test dataset. Here, the training size is kept at 80% whereas the testing size is kept at 20%. The following figure 23 and 24 shows the size of training and testing data respectively.

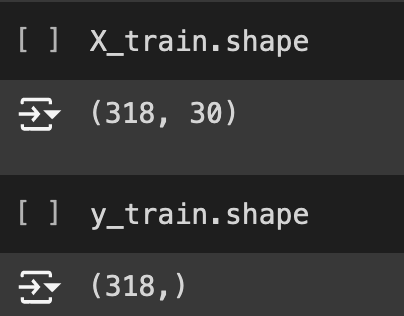


Figure : Size of Training Data for Feature and Target Dataset.

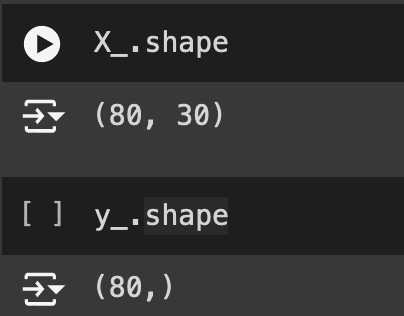


Figure 24: Size of Testing Data for Feature and Target Dataset.

1. Model Building:

The neural network model is developed using Keras API from TensorFlow. As shown by the following figure 25 sequential model is used, meaning that the layers will be added one after another in a sequence.

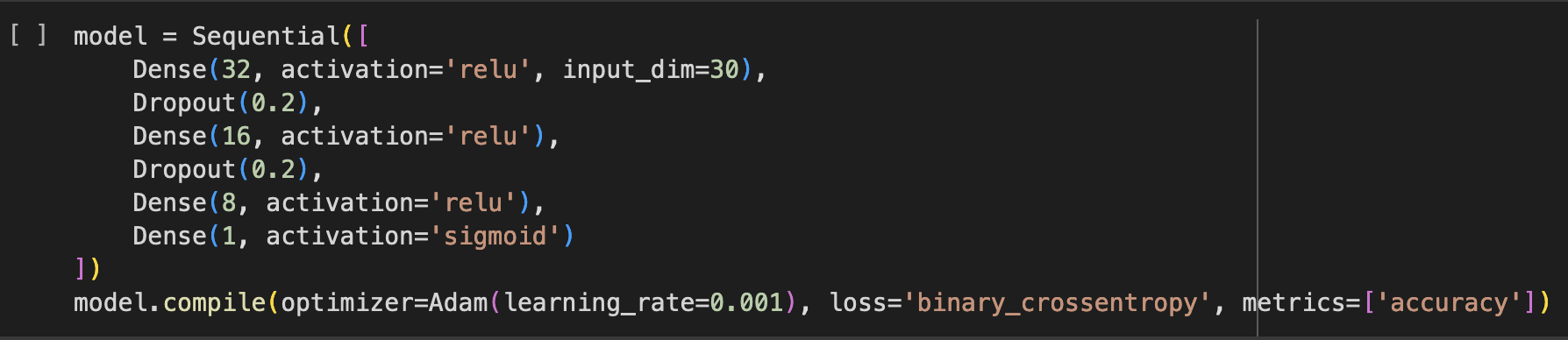


Figure : Building the Neural Network Model.

As shown by figure 25, the model has of three hidden dense layers. Each of the layer has ‘relu’ activation function which is used due to its non-linearity nature. Furthermore, the first dense layer starts with 32 neurons as it inputs all the columns of the dataset at once. Similarly, the second and third dense hidden layer have 16 and 8 neurons respectively. Each of the dense layers have less neurons in order to capture complex patterns by refining and consolidating the 32 features form first dense layer which helps in making more accurate prediction. Dropput layers are also mentioned in the model. In the dropout layer the rate of mitigating overfitting is kept at 20%. This is done in order to make sure that the model doesn’t relay heavily on certain neurons which have better correlation with the target column.

The output or the number of neurons for each of the layers and the parameters which is the sum of weight and biases for each of the layer.

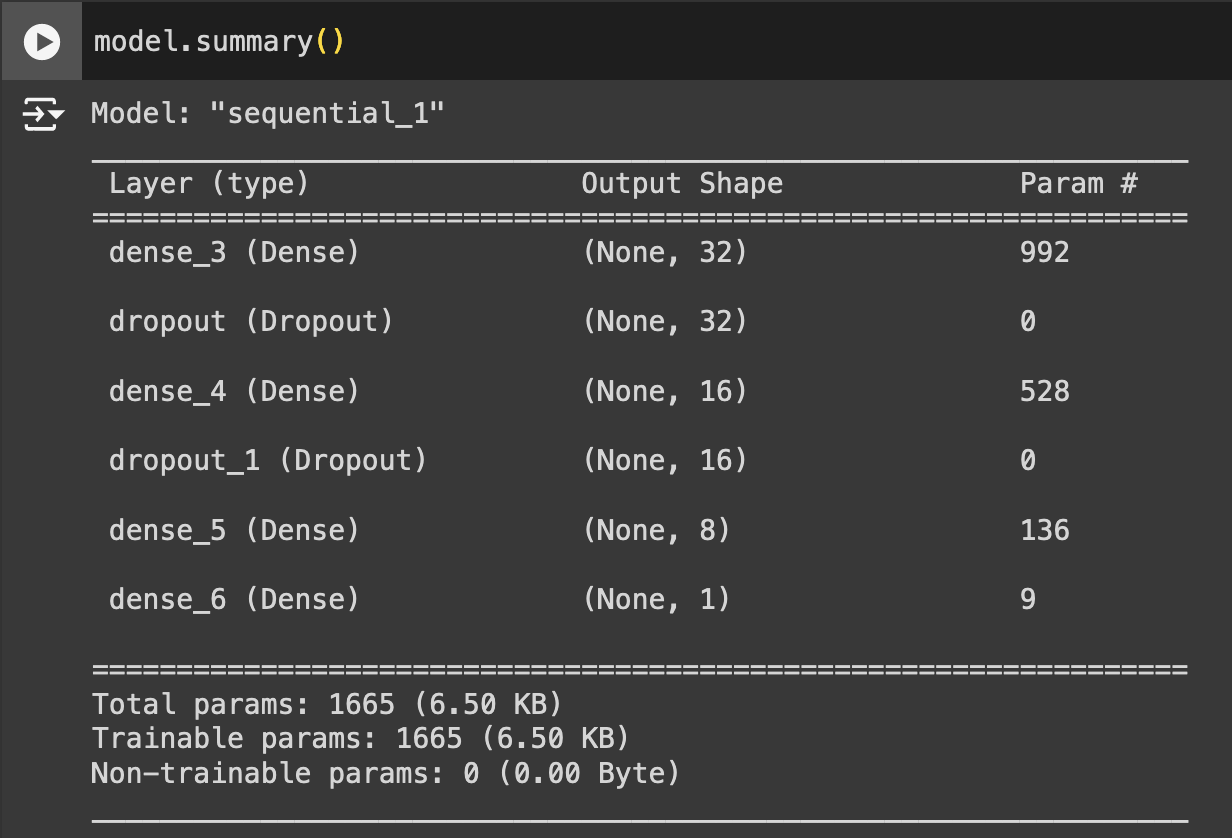


Figure : Summary of the Neural Network Model.

1. Model Training:

After the development of the model, the train dataset is fit into the model along with the validation dataset to cross check its performance. As shown by the following figure 27 the train dataset is fit into the model which was previously discussed. Further, the training is done for 200 epochs where each epochs signifies one complete pass through the entire training dataset. Likewise, each epoch contains a batch size of 9, meaning in one iteration of the model training 9 samples are processed before updating the weight and biases. Batch size is important as updating the weight and biases each of the sample would be a very rigorous work taking up much space whereas in a batch an average of the batch can be considered.

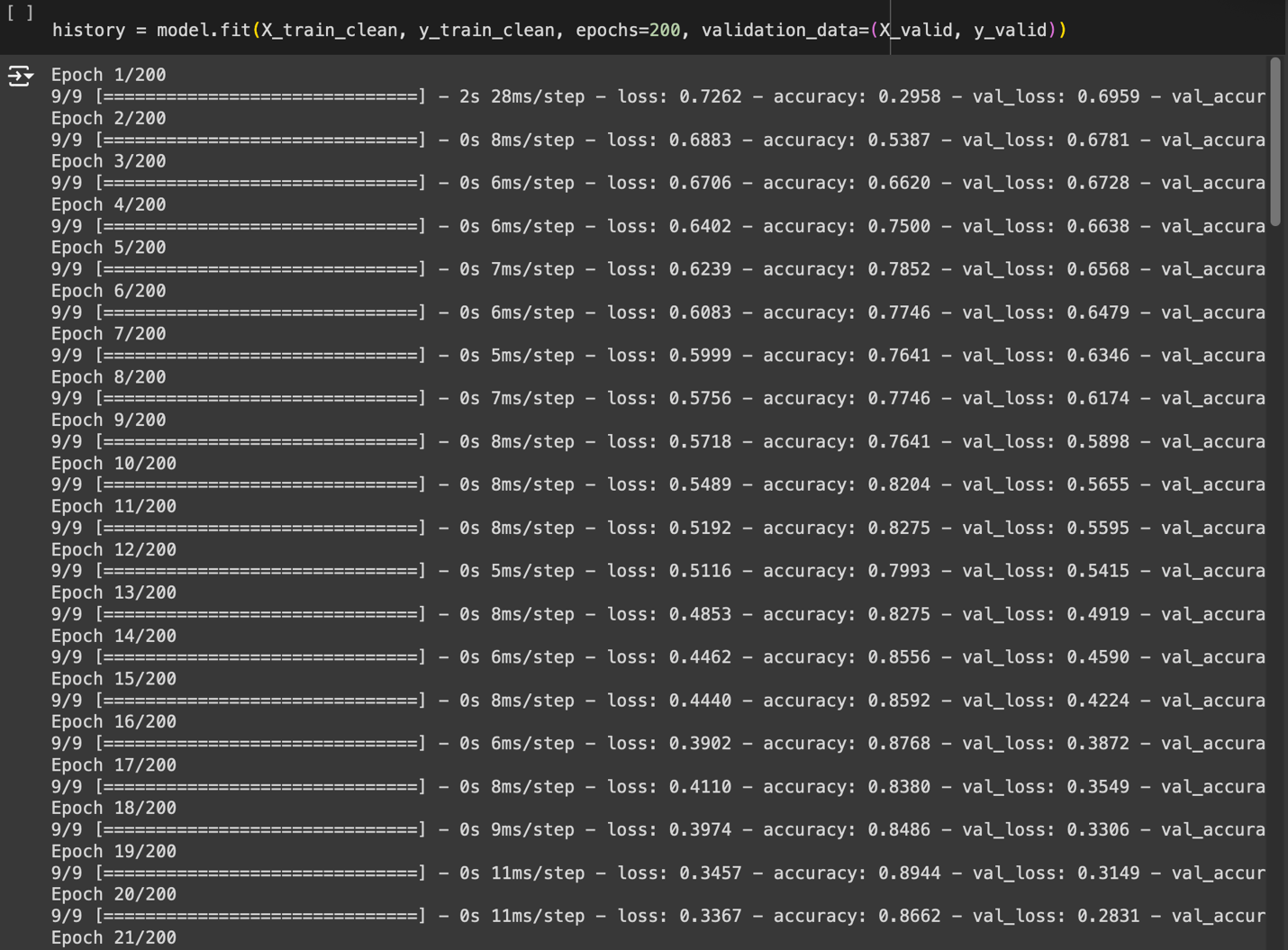


Figure : Training the Model.

In the figure 24, the result shows that the tanning accuracy at 98.6% and loss at 0.0130, which are the best accuracy and loss taken out form all the 200 ephocs. This outcome suggest that the model is learning patterns of training data very accurately also the low loss value indicates that the prediction is very close to the actual value. Similarly, the validation accuracy on the validation dataset also shows excellent performances the accuracy reaches to 100% and the loss at 0.0121. Since both the training and validation data are excellent this suggests that the model is most certainly not overfitting and is generalizing well towards unseen data.

1. Model Evaluation:

As shown by the following figure 28, a line graph for model accuracy and loss between the train and validation dataset. The blue line represents the train and orange line represents validation dataset. For the accuracy the train dataset gradually increased and reached the highest around 100 epochs whereas for the validation dataset the accuracy remained constant at around 50 epochs. Likewise for the loss same kind of patterns in downward trend can be observed for both training and validation dataset.

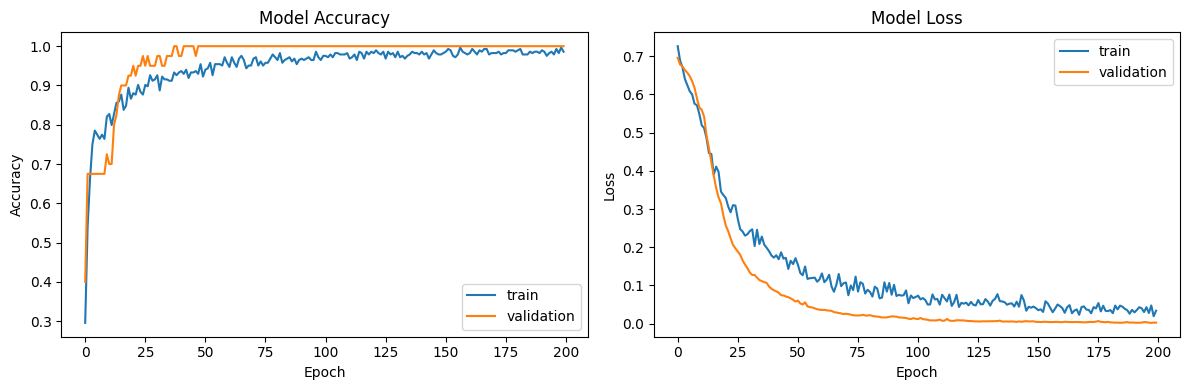


Figure : Plotting the Accuracy and Loss of the Model for Training and Validation Dataset.

Now the model will be evaluated based on the testing data by using the prediction model to the testing data and generating a confusion matrix which provides a report on accuracy, precision, recall, and F1-score which is shown by the following figure 29.

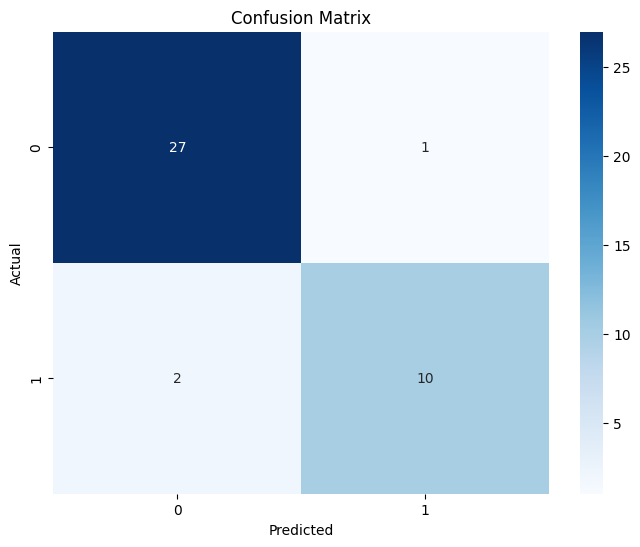


Figure : Confusion Matrix for the Model.

# Discussion of Findings

The dataset consisted of 569 rows and 33 columns out of which ‘diagnosis’ is a target column and the rest were feature columns. With initial the data exploration, the ‘unnamed:32’ column was found to be completely empty which was dropped for better accuracy.. Hence, out of the 30 columns 29 of them were found to have outliers after. Now the outliers were removed and the new record was 398 rows. This 398 rows had no outliers, as a result, it made the dataset more accurate. Similarly, with the help of correlation matrix the understating of relation between features and target was enhanced as the correlation varied for various feature. Finally, the neural network model was prepared with three hidden layers and tow dropout layer in order to prevent the model in memorizing certain patterns form the training dataset and predicting instead of generalizing. With the analysis of training and validation accuracy and loss through figure 27 and 28 the model performed exceptionally good where it generalized and signs of overfitting were not very prominent. To be precise the model achieved 98.6% and 100% accuracy with the training and validation respectively dataset in over 200 epochs. Likewise, the final evaluation also showed an accuracy of 93% with the unseen dataset.

# Conclusion

This project effectively showcased the use of neural network algorithms in diagnosing breast cancer using the Wisconsin Breast Cancer Dataset. The methodical approach encompassed critical steps such as data preprocessing, clustering, anomaly detection, and classification, culminating in the development of a robust model with high accuracy. The success of this model underscores the importance of a comprehensive data analysis pipeline in achieving reliable predictive performance. For future endeavors, exploring advanced neural network architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), could offer further enhancements. Additionally, applying the model to larger and more diverse datasets would be beneficial to validate and potentially improve its effectiveness, ensuring broader applicability and robustness in various clinical settings.

# References

American Cancer Society. (2021). Breast Cancer Facts & Figures 2021-2022. Atlanta: American Cancer Society, Inc.

Chae, Y. and Wilke, D.N. (2019) 'Empirical study towards understanding line search approximations for training neural networks', ArXiv, abs/1909.06893. Available at: <https://api.semanticscholar.org/CorpusID:202578064> [Accessed: 27 July 2024].

Chandola, V., Banerjee, A. and Kumar, V. (2009) 'Anomaly detection: A survey', ACM Computing Surveys, 41(3). Available at: <https://doi.org/10.1145/1541880.1541882> [Accessed: 1 August 2024].

Dua, D. and Graff, C. (2019) UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA. Available at: <http://archive.ics.uci.edu/ml> [Accessed: 27 July 2024].

Esteva, A., Kuprel, B., Novoa, R., et al. (2017) 'Dermatologist-level classification of skin cancer with deep neural networks', Nature, 542, pp. 115-118. Available at: <https://doi.org/10.1038/nature21056> [Accessed: 27 July 2024].

Guyon, I. and Elisseeff, A. (2003) 'An introduction to variable and feature selection', Journal of Machine Learning Research Special Issue on Variable and Feature Selection, 3, pp. 1157-1182. Available at: <https://https://doi.org/10.1162/153244303322753616> [Accessed: 1 August 2024].

Provost, F. and Fawcett, T. (2013) 'Data science and its relationship to big data and data-driven decision making', Big Data, 1(1). Available at: <https://https://doi.org/10.1145/1541880> [Accessed: 1 August 2024].

Sidey-Gibbons, J.A.M. and Sidey-Gibbons, C.J. (2019) 'Machine learning in medicine: a practical introduction', BMC Medical Research Methodology, 19. Available at: [https://api.semanticscholar.org/CorpusID:84183143](https://api.semanticscholar.org/CorpusID:84183143" \t "_new) [Accessed: 27 July 2024].

Tanaka, F. and Aranha, C. de C. (2019) 'Data Augmentation Using GANs', ArXiv, abs/1904.09135. Available at: <https://api.semanticscholar.org/CorpusID:126180363> [Accessed: 27 July 2024].

Van Looveren, A. and Klaise, J. (2019) 'Interpretable Counterfactual Explanations Guided by Prototypes', ArXiv, abs/1907.02584. Available at: <https://api.semanticscholar.org/CorpusID:195820639> [Accessed: 27 July 2024].

# Appendices

# Importing Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder, MinMaxScaler

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

from tensorflow.keras.optimizers import Adam

from sklearn.metrics import confusion\_matrix, classification\_report, mean\_squared\_error

from sklearn.cluster import KMeans

from sklearn.decomposition import PCA

from sklearn.neighbors import LocalOutlierFactor

from google.colab import files

uploaded = files.upload()

# Reading Dataset

# Load data

df = pd.read\_csv('data.csv')

# Exploring Data

df.shape

df.head()

df.tail()

df.info()

df.describe()

df.isnull().sum()

df.duplicated().sum()

# Cleaning Data

df.drop(columns=['id','Unnamed: 32'],inplace=True)

\*\*Checking For Outliers\*\*

columns = df.columns

for column in columns:

if df[column].dtype == 'float64':

fig, (ax\_box, ax\_hist) = plt.subplots(1, 2, figsize=(10, 5))

ax\_box.boxplot(df[column], vert=False, whis=1.5)

ax\_box.set\_xlabel('Value')

ax\_box.set\_title(f'{column} Box')

sns.histplot(df[column], bins=50, color='blue', kde=True, ax=ax\_hist)

ax\_hist.set\_xlabel('Value')

ax\_hist.set\_ylabel('Frequency')

ax\_hist.set\_title(f'{column} dist')

# Show plot

plt.tight\_layout()

plt.show()

\*\*Dealing With Outliers\*\*

filter\_data=pd.DataFrame()

for column in columns:

if df[column].dtype=='float64':

filter\_data[column]=df[column]

print(filter\_data.info())

for column in filter\_data.columns:

Q1 = filter\_data[column].quantile(0.25)

Q3 = filter\_data[column].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

df[column] = filter\_data[(filter\_data[column] >= lower\_bound) & (filter\_data[column] <= upper\_bound)][column]

df.dropna(inplace=True)

df.info()

\*\*Showing Outliers After Dealing With It\*\*

columns = df.columns

for column in columns:

if df[column].dtype == 'float64':

fig, (ax\_box, ax\_hist) = plt.subplots(1, 2, figsize=(10, 5))

ax\_box.boxplot(df[column], vert=False, whis=1.5)

ax\_box.set\_xlabel('Value')

ax\_box.set\_title(f'{column} Box')

sns.histplot(df[column], bins=50, color='blue', kde=True, ax=ax\_hist)

ax\_hist.set\_xlabel('Value')

ax\_hist.set\_ylabel('Frequency')

ax\_hist.set\_title(f'{column} dist')

# Show plot

plt.tight\_layout()

plt.show()

# Preprocessing

df

\*\*Splitting Data to X and y\*\* (X contains all features and y contains Target)

X = df.drop('diagnosis',axis=1)

y = df['diagnosis']

\*\*Using Label Encoder\*\*

encoder = LabelEncoder()

y = encoder.fit\_transform(y)

X

y

\*\*Normalizing Data Using MinMax Scaler\*\*

minmaxscaler = MinMaxScaler()

X = minmaxscaler.fit\_transform(X)

X

newdf = df.drop('diagnosis',axis=1)

correlation\_matrix = newdf.corr()

plt.figure(figsize=(20, 10))

heatmap = sns.heatmap(correlation\_matrix,

annot=True,

fmt=".2f",

cmap='coolwarm',

linewidths=0.5,

cbar\_kws={"shrink": .8})

plt.title('Heatmap of Feature Correlations')

plt.show()

# Splitting Data into Train and Test

X\_train, X\_, y\_train, y\_ = train\_test\_split(X, y, test\_size=0.20, random\_state=42)

X\_valid, X\_test, y\_valid, y\_test = train\_test\_split(X\_, y\_, test\_size=0.5, random\_state=42)

X\_train.shape

y\_train.shape

X\_.shape

y\_.shape

X\_valid.shape

y\_valid.shape

# Clustering

kmeans = KMeans(n\_clusters=2, random\_state=42)

clusters = kmeans.fit\_predict(X)

\*\*Dimensionality Reduction for Visualization\*\*

pca = PCA(n\_components=2)

pca\_result = pca.fit\_transform(X)

plt.scatter(pca\_result[:, 0], pca\_result[:, 1], c=clusters, cmap='viridis')

plt.title('PCA of Clusters')

plt.xlabel('PCA Component 1')

plt.ylabel('PCA Component 2')

plt.show()

# Anomaly Detection

lof = LocalOutlierFactor(n\_neighbors=20, contamination=0.1)

outliers = lof.fit\_predict(X)

outlier\_indices = np.where(outliers == -1)[0]

\*\*Remove Anomalies From the Training Set\*\*

outlier\_indices = [idx for idx in outlier\_indices if idx < X\_train.shape[0]]

X\_train\_clean = np.delete(X\_train, outlier\_indices, axis=0)

y\_train\_clean = np.delete(y\_train, outlier\_indices, axis=0)

# Build the Model

model = Sequential([

Dense(32, activation='relu', input\_dim=30),

Dropout(0.2),

Dense(16, activation='relu'),

Dropout(0.2),

Dense(8, activation='relu'),

Dense(1, activation='sigmoid')

])

model.compile(optimizer=Adam(learning\_rate=0.001), loss='binary\_crossentropy', metrics=['accuracy'])

model.summary()

history = model.fit(X\_train\_clean, y\_train\_clean, epochs=200, validation\_data=(X\_valid, y\_valid))

y\_pred = (model.predict(X\_test) > 0.5).astype(int)

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

# Plot accuracy

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='train')

plt.plot(history.history['val\_accuracy'], label='validation')

plt.title('Model Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

# Plot loss

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label='train')

plt.plot(history.history['val\_loss'], label='validation')

plt.title('Model Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend()

plt.tight\_layout()

plt.show()

# Confusion Matrix

y\_pred = model.predict(X\_test)

y\_pred\_classes = (y\_pred > 0.5).astype("int32")

# Confusion matrix

cm = confusion\_matrix(y\_test, y\_pred\_classes)

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()