BREAST CANCER

ANALYSIS AND DETECTION USING MACHINE LEARNING

PROJECT GUIDE

PROF. BHAVANA SAMUDRA

PROJECT MEMBERS:

KALPESH TAPASE

SHUBHAM BHIDE

KIMAYA AHER

KETAKI SAHASRABUDHE

SHUBHAM GALANDE

PRADIP SIRSAT

ACKNOWLEDGEMENT

We would like to thank the Statistics department, Sir Parshurambhau College for giving us this opportunity to carry out this project which has helped us in a better understanding of the subject.

We would also like to thank our mentor Bhavana Ma’am for helping and guiding us along the process.

Most of all we would like to thank all the people who guided us throughout our journey.

Contents:

Introduction:

Cancer is a group of diseases involving abnormal cell growth with the potential to invade or spread to other parts of the body. These contrast with tumours, which do not spread.

Cancers comprise a large family of diseases that involve abnormal cell growth with the potential to invade or spread to other parts of the body. They form a subset of neoplasms. A neoplasm or '**Tumour" is a group of cells that have undergone unregulated growth and will often form a mass or lump, but may be distributed diffusely.**

The tumour can be benign or malignant.

BENIGN TUMOUR:

- A **benign tumour** is a mass of cells (tumour) that does not invade neighbouring tissue or metastasize (spread throughout the body). Compared to tumours, benign tumours generally have a slower growth rate.
- Benign tumours have relatively well-differentiated cells.
- They are often surrounded by an outer surface (fibrous sheath of connective tissue) or stay contained within the epithelium

MALIGNANT TUMOUR:

- A **malignant** tumour contrasts with a non-cancerous in that a malignancy is not self-limited in its growth, is capable of invading adjacent tissues, and may be capable of spreading to distant tissues.
- When malignant cells form, symptoms do not typically appear until there has been a significant growth of the mass.
- Once signs and symptoms do arise, they are dependent on the location, size and type of malignancy.

Possible signs and symptoms include a lump, abnormal bleeding, prolonged cough, unexplained weight loss, and a change in bowel movements. While these symptoms may indicate cancer, they can also have other causes. Over 100 types of cancers affect humans.

The most common types of cancer in males are lung cancer, prostate cancer, colorectal cancer, and stomach cancer.

"In females, the most common types are

BREAST CANCER, colorectal cancer, lung cancer, and cervical cancer."

Breast cancer is a cancer that develops from breast tissue.

Signs of breast cancer may include a lump in the breast, a change in breast shape, a dumpling of the skin, milk rejection, fluid coming from the nipple, a newly inverted nipple, or a red or scaly patch of skin.

In those with a distant spread of the disease, there may be bone pain, swollen lymph nodes, shortness of breath, or yellow skin.

Breast cancer, like other cancers, occurs because of an interaction between an environmental (external) factor and a genetically susceptible host.

Normal cells divide as many times as needed and stop. They attach to other cells and stay in place in tissues. Cells become cancerous when they lose their ability to stop dividing, to attach to other cells, to stay where they belong, and to die at the proper time.

Cancer is a worldwide epidemic that affects individuals of all ages and backgrounds. There are many types of cancer, however, breast cancer is one of the most common cancers in women.

Due to this challenge, researchers should pay special attention to cancer detection and prognosis.

"Predicting and diagnosing cancer at an early stage is an area where machine-learning approaches may have a significant impact."

Objectives:

- Establishing an adequate model by revealing the predictive factors of early-stage breast cancer patients
- A more comprehensive comparison and analysis using data visualization and machine learning applications for breast cancer detection and visibility to validate the model
- Observe which features are most effective in predicting breast cancer
- Evaluating the performance of different machine learning algorithms on the given data

Data Description:

The Breast Cancer Wisconsin Diagnostic dataset was initially obtained from the University of Wisconsin Hospitals, Madison, by Dr. William H. Wolberg.

It was donated to the UCI Machine Learning Repository, where it has been widely used for research and educational purposes.

The dataset typically consists of 569 instances, each representing a tissue sample from a patient.

Each instance has 30 features computed from digitized images of fine needle aspirates (FNAs) of breast mass tissue samples.

Here is the attribute information for the Breast Cancer Wisconsin (Diagnostic) Data Set:

1. **ID**: The unique identification number for each instance (patient).
2. **Diagnosis (Class)**: The diagnosis of breast tissues, which is the target variable. It is categorized into two classes:
    - Malignant (M): Representing cancerous tumours.
    - Benign (B): Representing non-cancerous tumours.
3. **Ten real-valued features are computed for each cell nucleus**:
    - **Radius (mean, se, worst)**: Mean of distances from the centre to points on the perimeter, standard error, and worst (largest mean value) of the mean of distances from the centre to points on the perimeter.
    - **Texture (mean, se, worst)**: Standard deviation of grey-scale values, standard error, and worst of grey-scale values.
    - **Perimeter (mean, se, worst)**: Mean of the perimeter, standard error, and worst of the perimeter.
    - **Area (mean, se, worst)**: Mean of area, standard error, and worst of the area.
    - **Smoothness (mean, se, worst)**: Mean of local variation in radius lengths, standard error, and worst of local variation in radius lengths.
    - **Compactness (mean, se, worst)**: Mean of perimeter^2 / area - 1, standard error, and worst of perimeter^2 / area - 1.
    - **Concavity (mean, se, worst)**: Mean of severity of concave portions of the contour, standard error, and worst of severity of concave portions of the contour.

- **Concave points (mean, se, worst)**: Mean of number of concave portions of the contour, standard error, and worst of number of concave portions of the contour.
- **Symmetry (mean, se, worst)**: Mean of symmetry, standard error, and worst of symmetry.
- **Fractal dimension (mean, se, worst)**: Mean of "coastline approximation" - 1, standard error, and worst of "coastline approximation" - 1.

The **30 features** represent various measurements extracted from the images of cell nuclei present in the breast tissue samples

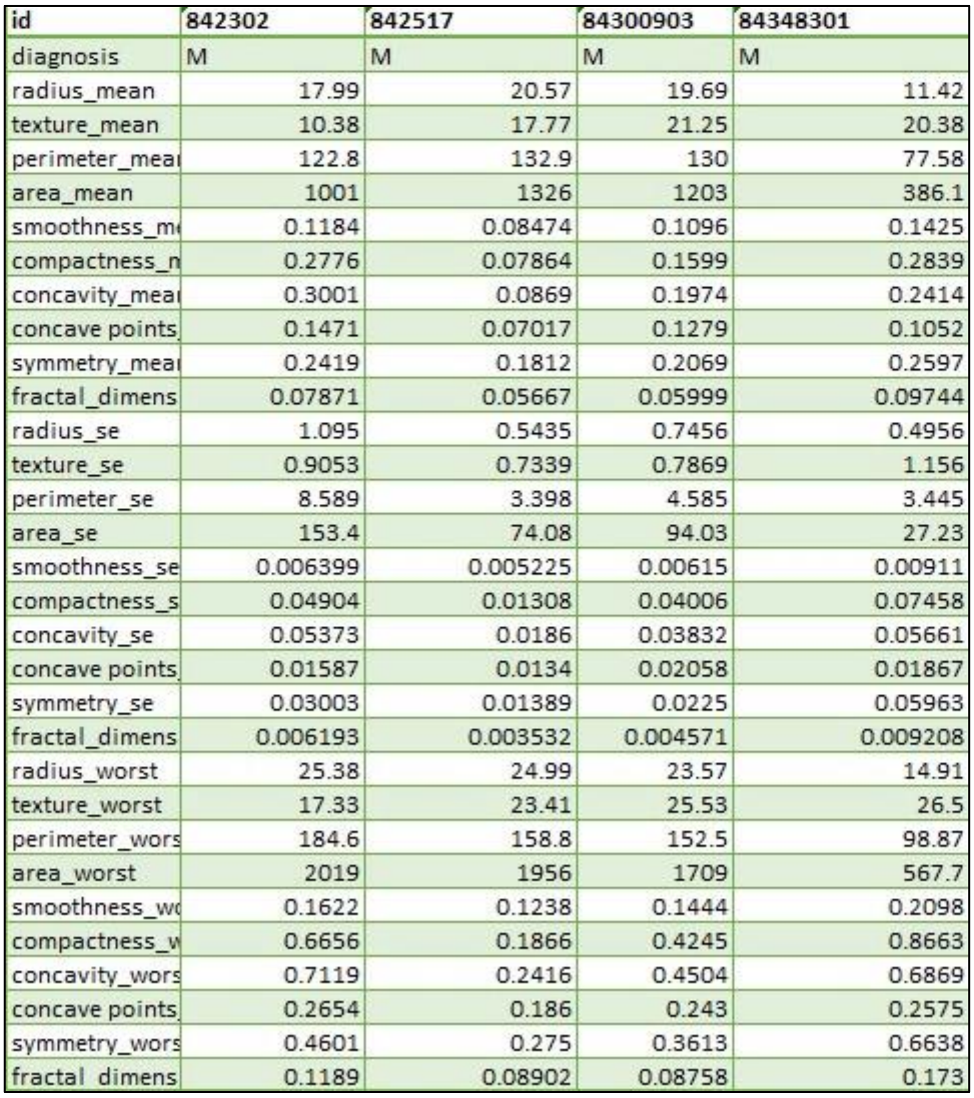
The target variable is the diagnosis of breast tissues, which can be either malignant (cancerous) or benign (non-cancerous). The diagnosis is indicated by the 'M' or 'B' class labels, respectively.

The dataset is publicly available and can be accessed from various sources, including the UCI Machine Learning Repository and Kaggle.

Data Overview:

The dataset contains 569 observations and 31 variables.
The variables include features such as radius, texture, perimeter, area, smoothness, compactness, concavity, symmetry, and fractal dimension, among others.
The target variable is "diagnosis," which indicates whether a tumour is malignant (M) or benign (B).

<u>**Data Cleaning:**</u>
The initial dataset contained 33 columns, including an "id" column and an unnamed column (probably an index), which were removed. There were no missing values in the dataset, as **confirmed by the sum (is. na(d)) command.**
This is an overview of the data set used.



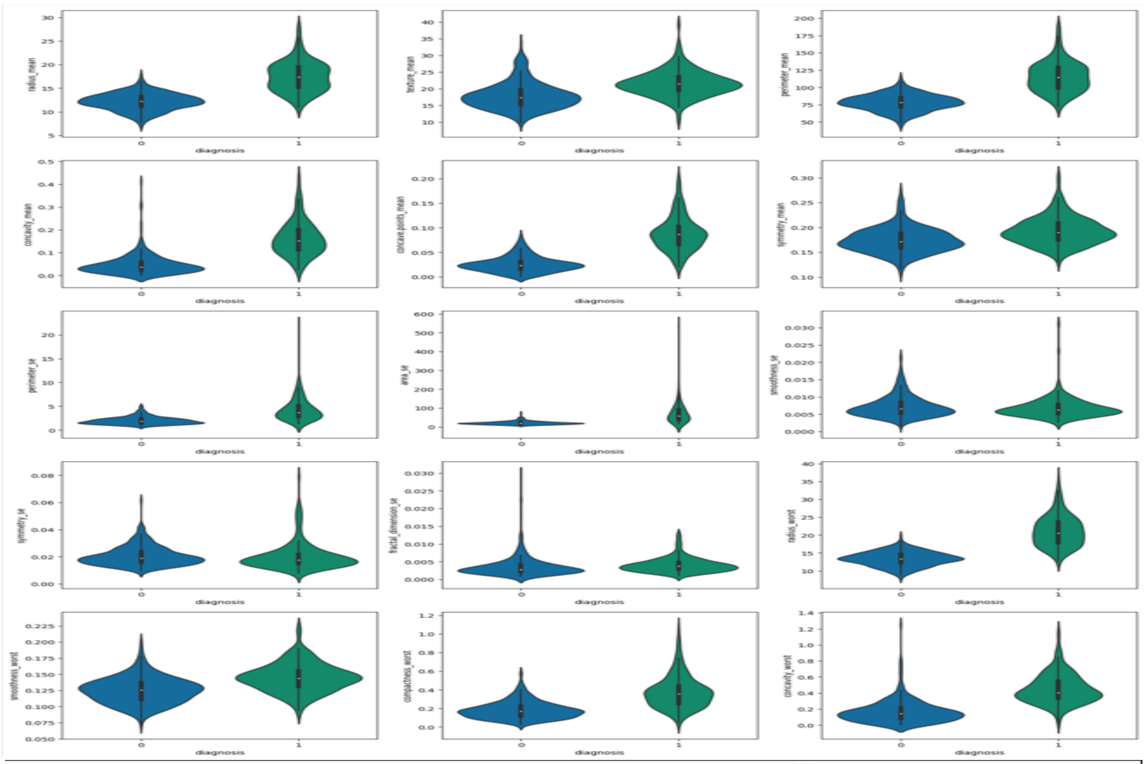
id	842302	842517	84300903	84348301
diagnosis	M	M	M	M
radius_mean	17.99	20.57	19.69	11.42
texture_mean	10.38	17.77	21.25	20.38
perimeter_mear	122.8	132.9	130	77.58
area_mean	1001	1326	1203	386.1
smoothness_m	0.1184	0.08474	0.1096	0.1425
compactness_n	0.2776	0.07864	0.1599	0.2839
concavity_mear	0.3001	0.0869	0.1974	0.2414
concave points	0.1471	0.07017	0.1279	0.1052
symmetry_mear	0.2419	0.1812	0.2069	0.2597
fractal_dimens	0.07871	0.05667	0.05999	0.09744
radius_se	1.095	0.5435	0.7456	0.4956
texture_se	0.9053	0.7339	0.7869	1.156
perimeter_se	8.589	3.398	4.585	3.445
area_se	153.4	74.08	94.03	27.23
smoothness_se	0.006399	0.005225	0.00615	0.00911
compactness_s	0.04904	0.01308	0.04006	0.07458
concavity_se	0.05373	0.0186	0.03832	0.05661
concave points	0.01587	0.0134	0.02058	0.01867
symmetry_se	0.03003	0.01389	0.0225	0.05963
fractal_dimens	0.006193	0.003532	0.004571	0.009208
radius_worst	25.38	24.99	23.57	14.91
texture_worst	17.33	23.41	25.53	26.5
perimeter_wors	184.6	158.8	152.5	98.87
area_worst	2019	1956	1709	567.7
smoothness_wo	0.1622	0.1238	0.1444	0.2098
compactness_w	0.6656	0.1866	0.4245	0.8663
concavity_wors	0.7119	0.2416	0.4504	0.6869
concave points	0.2654	0.186	0.243	0.2575
symmetry_wors	0.4601	0.275	0.3613	0.6638
fractal_dimens	0.1189	0.08902	0.08758	0.173

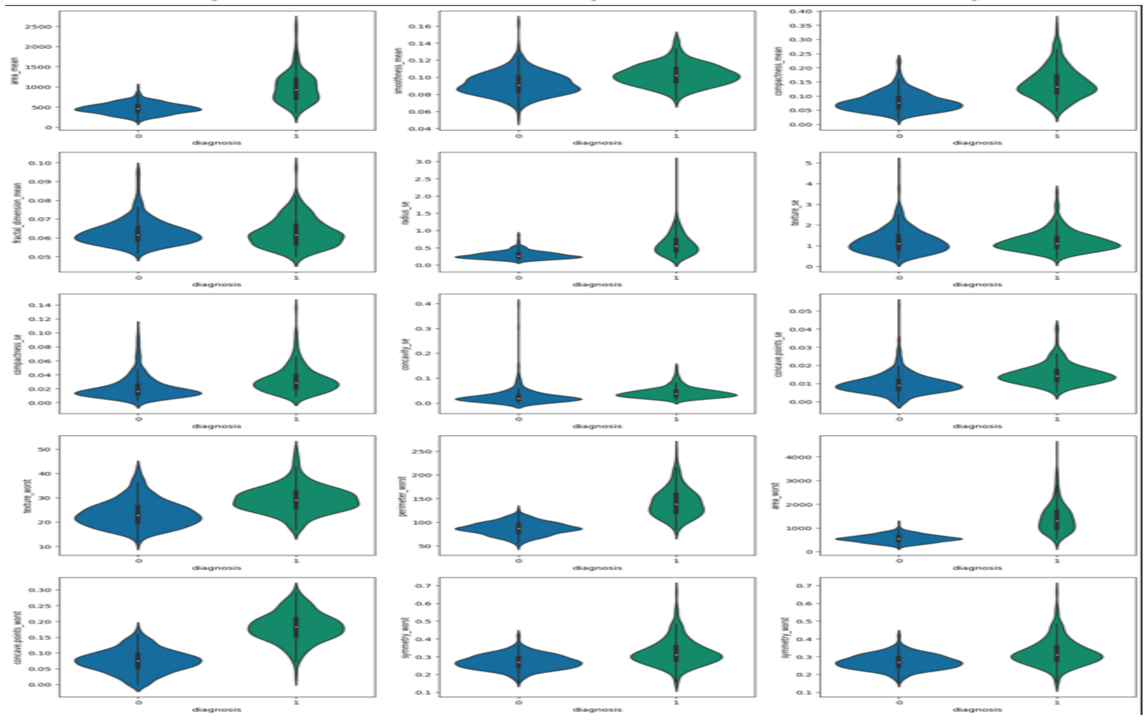
Exploratory Data Analysis:

Summary statistics provide insights into the distribution, central tendency, and spread of each variable.

Violin Plot-

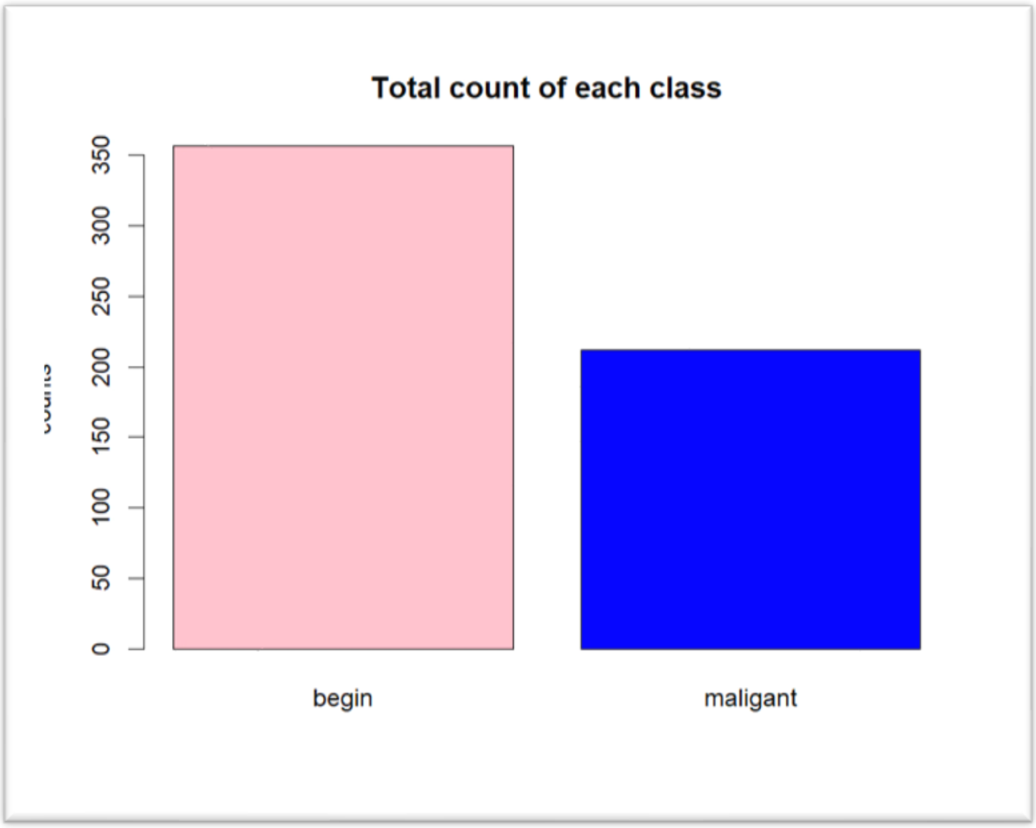
This analysis aims to understand the distribution of these variables across the two classes and to identify any patterns or differences that may exist between them. To achieve this, violin plots are employed as they provide a visual representation of the data distribution, allowing for easy comparison between the classes.





Interpretations from violin plot:
The use of violin plots has provided valuable insights into the distribution of variables across benign and malignant classes. By visually comparing the distributions, we can identify variables that exhibit clear differences between the two classes such as the variables area_worst, smoothness_mean, perimeter_worst etc, as well as those that may not be as informative in distinguishing between them. This gives the first insight into variable-importance decision-making techniques.

Visualization:
A bar plot was created to visualize the distribution of the target classes ("M" for malignant and "B" for benign). This helps to understand the balance or imbalance between the classes.



Data Scaling:
The data was scaled using the 'scale ()' function to standardize the features, which is a common preprocessing step before applying machine learning algorithms.
Scaling ensures that all features contribute equally to the analysis, preventing features with larger scales from dominating the model.

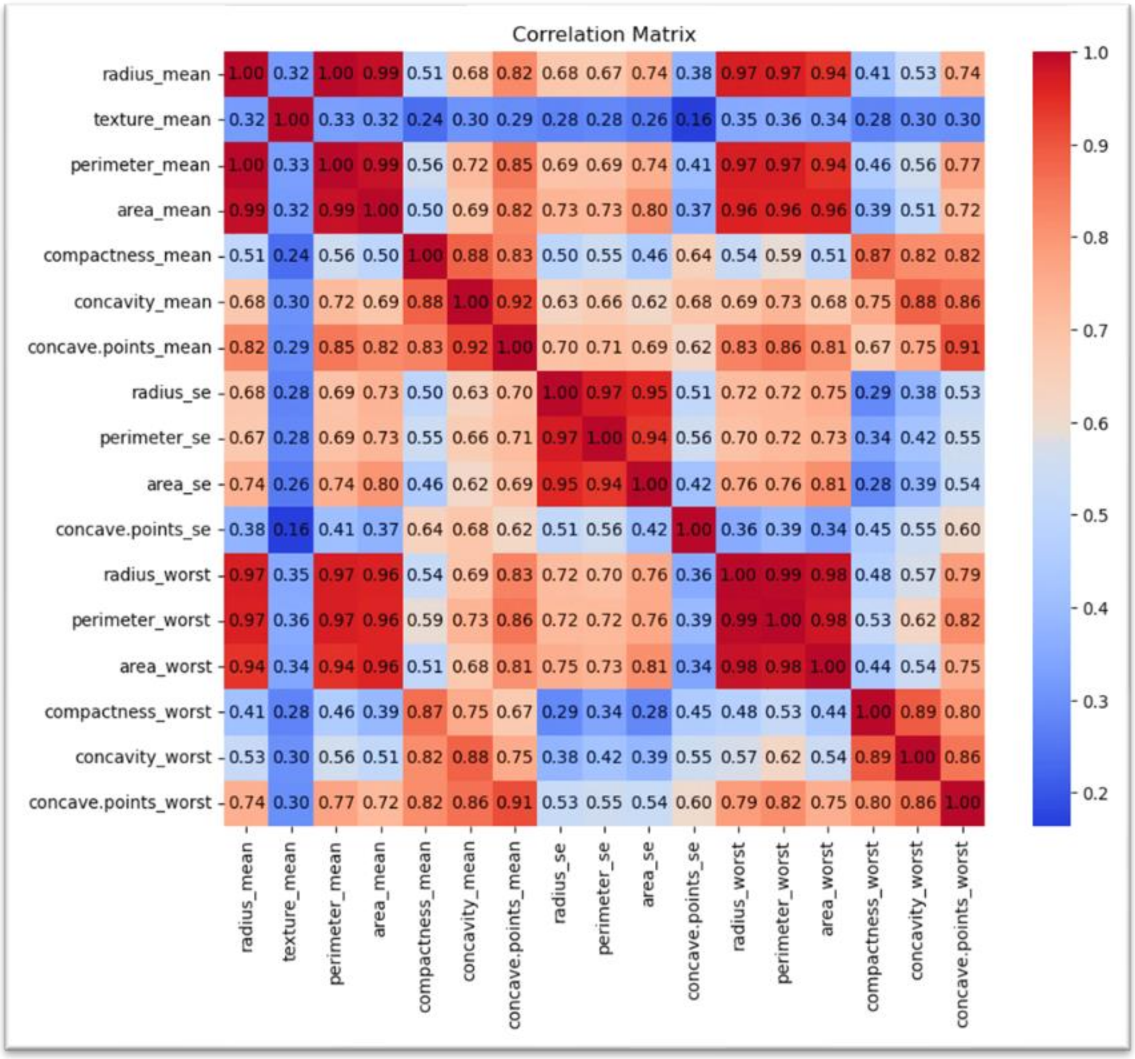
Variable Encoding:

The target variable "diagnosis" was encoded into numeric values (1 for benign and 2 for malignant), possibly for compatibility with machine learning algorithms.

Correlation Analysis:

The mean correlation coefficient between all variables was calculated and found to be approximately 0.3965. The given heatmap is only for variables having a significant and considerable correlation

This indicates a moderate positive correlation among the variables correlating higher than 0.3



Heat Map

The preprocessing steps, including data cleaning, scaling, and encoding, have prepared the dataset for further analysis, such as model building or feature selection.
The dataset appears to be well-prepared for applying machine learning algorithms to predict breast cancer diagnosis based on tumour characteristics.
Overall, the preprocessing steps have provided a clean and standardized dataset ready for further analysis, such as predictive modelling or exploratory data analysis.

Concepts Used:
Random forest for feature importance, PCA, Clustering, K-means clustering, model fitting, Naïve Bayes, KNN, Logistic regression, scree plot, F1 score.

Random forest for feature importance

Random Forest constructs multiple decision trees on random subsets of the data and aggregates their predictions to make final classifications. During this process, each tree's performance is evaluated, and variable importance scores are calculated based on how much each predictor improves the model's accuracy.
The concept of variable importance in Random Forest is derived from the decrease in prediction accuracy when a particular variable is randomly permuted. This measure, known as Mean Decrease Accuracy, quantifies the impact of each predictor on model performance. Variables with higher importance scores contribute more significantly to reducing prediction error across the ensemble of trees.
By evaluating variable importance scores, it identifies the most informative predictors, facilitating model interpretability, and improving predictive performance.
Why Random Forest?
Random Forest is less susceptible to overfitting compared to single decision trees, making it suitable for datasets with noisy or high-dimensional features.

PCA (Principal Component Analysis)

PCA is a statistical technique used for dimensionality reduction in data analysis. PCA identifies patterns in data, expressing it in a way that highlights its similarities and differences.

It does this by transforming the data into a new coordinate system, where the axes (principal components) are orthogonal and ordered by the amount of variance they explain. This makes it useful for simplifying complex datasets while retaining important information.

Logistic Regression

Logistic regression is a technique that was first used for biological studies in the early twentieth century. It has become widespread in social studies too. Logistic regression is also one of the predictive analyses. Logistic regression is appropriate to use when there is one binary dependent variable and other independent variables. Linear and logistic regressions are different in terms of the dependent variable.

LOGISTIC REGRESSION MODEL:

In Statistics, the logistic model is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1 and the sum adding to one.

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable.

Formula:

$\Pi(X)= (\exp(\beta0 + \beta1x1 + \beta2\ x2 + \dots\ \beta n\ Xn)) / (1+ \exp(\beta\ 0 + \beta\ 1\ x\ 1 + \beta\ 2\ x\ 2 + \dots\ \beta n\ Xn))$

$\Pi(X) = 1$ Probability of success & $\Pi(X)= 0$ Probability of failure

Where, when the value of $\Pi(X)$ is greater than 0.5 is considered a success and vice versa.

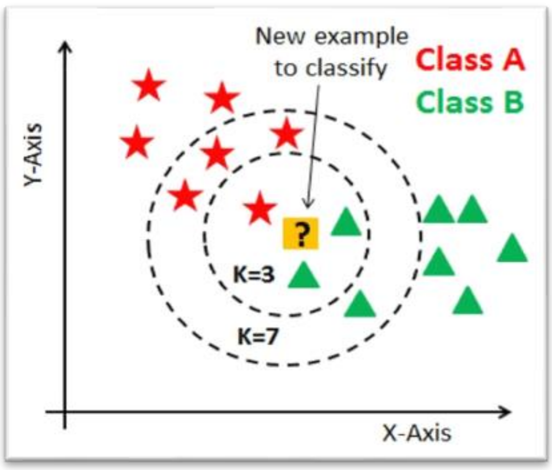
K-Nearest Neighbour (KNN)

KNN is a supervised learning technique that means the label of the data is identified before making predictions. Clustering and regression are two purposes to use it.

K represents a numerical value for the nearest neighbours. The KNN algorithm does not have a training phase.

Predictions are made based on the Euclidean distance to k-nearest neighbours.

The label is classified according to the nearest neighbour to the class labels of its neighbours.



Naïve Bayes

Naïve Bayes is a straightforward and also fast algorithm for classification. Its working process is based on Bayes theorem. It is represented below:

$$P(X|Y) = P(Y|X) * P(X)P(Y)$$

Naive Bayes assumes that the presence of a particular feature in a class is independent of the presence of any other feature. This is a strong assumption and hence the term "naive".

Naive Bayes calculates the probability of each class given the input features using Bayes' theorem and the assumption of feature independence. It selects the class with the highest probability as the prediction.

Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows visualization of the performance of an algorithm, particularly in binary classification problems, where there are two possible outcomes. Here's how a confusion matrix is structured:

True Positive (TP): These are the cases where the model correctly predicts the positive class.
True Negative (TN): These are the cases where the model correctly predicts the negative class.
False Positive (FP): Also known as a Type I error, these are the cases where the model incorrectly predicts the positive class when it's negative.
False Negative (FN): Also known as a Type II error, these are the cases where the model incorrectly predicts the negative class when it's positive.

F1 Score

The F1 score is a metric commonly used in binary classification tasks to evaluate the performance of a classifier. It takes into account both the precision and recall of the classifier and provides a single score that balances between them.

Precision (also called Positive Predictive Value): Precision measures the proportion of true positive predictions among all positive predictions made by the classifier.

Recall (also called Sensitivity or True Positive Rate): Recall measures the proportion of true positive predictions among all actual positive instances in the dataset.

F1 Score: The F1 score is the harmonic mean of precision and recall. The harmonic mean gives more weight to lower values. Therefore, **the F1 score will be high only if both precision and recall are high.**

Precision = True Positives / (True Positives + False Positives)
Recall =True Positives/ (True Positives + False Negatives)
F1 =2 (Precision× Recall)/ (Precision + Recall)

The F1 score ranges from 0 to 1, where a higher value indicates better performance. A perfect classifier would have an F1 score of 1, while a completely random classifier would have an F1 score of 0.

Scree Plot

The scree plot is a line plot where the x-axis represents the factor or component number, and the y-axis represents the corresponding eigenvalues. The plot typically shows a steep drop in eigenvalues initially, followed by a levelling off. The point at which the eigenvalues level off is often referred to as the "elbow" of the plot.

The scree plot helps in determining the optimal number of factors or components to retain in the analysis.

Clustering and K-means clustering

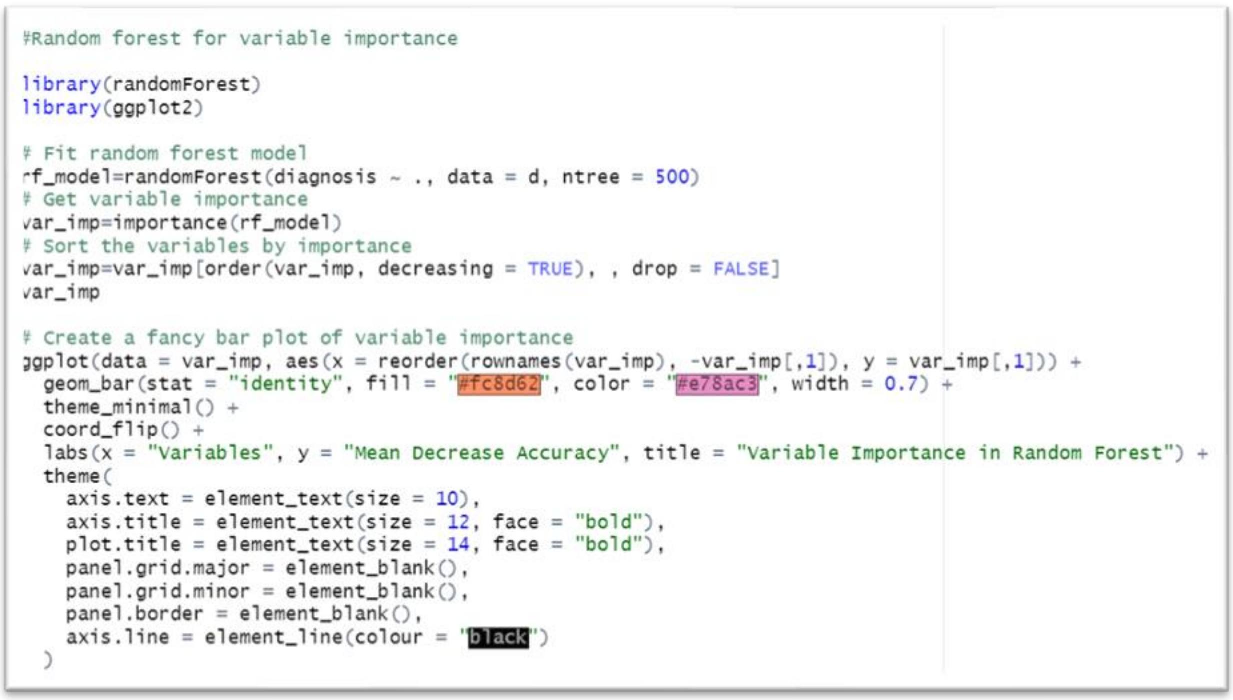
Clustering is a fundamental technique in unsupervised machine learning used to group similar data points into clusters or segments. It's a form of exploratory data analysis that aims to find natural groupings or patterns in the data without any prior knowledge of the labels or classes.

K-means clustering is one of the most popular and widely used clustering algorithms. It's a partition-based clustering algorithm that aims to divide a dataset into k clusters, where each data point belongs to the cluster with the nearest mean (centroid).

Selecting the appropriate number of clusters (k) is crucial in k-means clustering. Common methods for determining k include the elbow method, silhouette score, and gap statistics. For each data point, k-means assigns it to the cluster with the nearest centroid. Analysing the cluster membership can help understand which data points are similar to each other and how they group tog

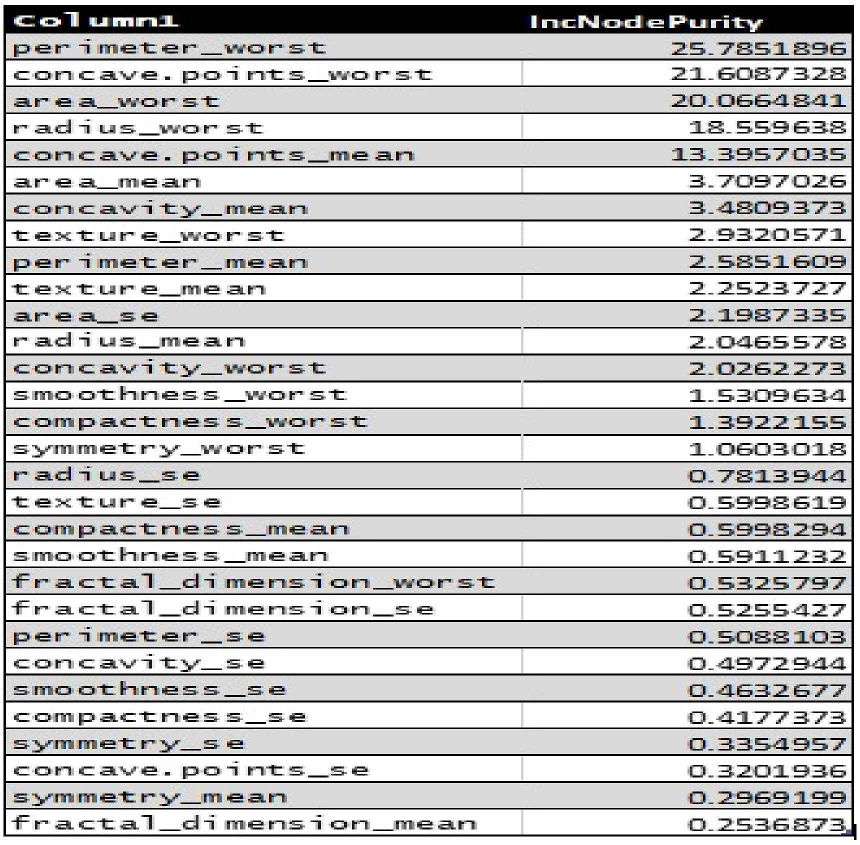
Random forest for feature importance

To determine the most significant variables from our dataset (variables that will impact our predictions the most), we use the random forest technique.



```
#Random forest for variable importance

library(randomForest)
library(ggplot2)

# Fit random forest model
rf_model=randomForest(diagnosis ~ ., data = d, ntree = 500)
# Get variable importance
var_imp=importance(rf_model)
# Sort the variables by importance
var_imp=var_imp[order(var_imp, decreasing = TRUE), , drop = FALSE]
var_imp

# Create a fancy bar plot of variable importance
ggplot(data = var_imp, aes(x = reorder(rownames(var_imp), -var_imp[,1]), y = var_imp[,1])) +
  geom_bar(stat = "identity", fill = "#fc8d62", color = "#e78ac3", width = 0.7) +
  theme_minimal() +
  coord_flip() +
  labs(x = "Variables", y = "Mean Decrease Accuracy", title = "Variable Importance in Random Forest") +
  theme(
    axis.text = element_text(size = 10),
    axis.title = element_text(size = 12, face = "bold"),
    plot.title = element_text(size = 14, face = "bold"),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    axis.line = element_line(colour = "black")
  )
```

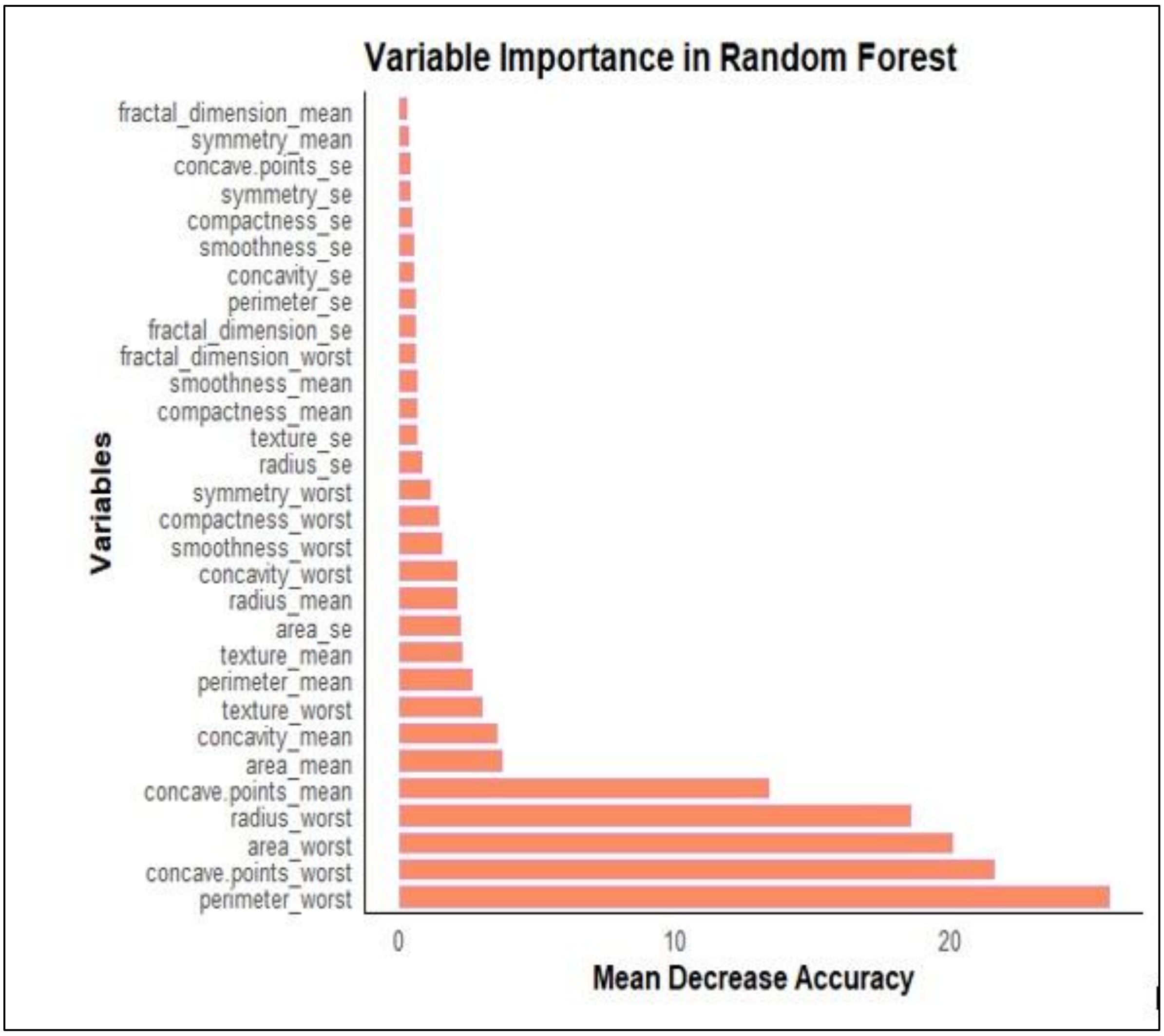
From this, the following are the variables and importance.



Column1	IncNodePurity
perimeter_worst	25.7851896
concave.points_worst	21.6087328
area_worst	20.0664841
radius_worst	18.559638
concave.points_mean	13.3957035
area_mean	3.7097026
concavity_mean	3.4809373
texture_worst	2.9320571
perimeter_mean	2.5851609
texture_mean	2.2523727
area_se	2.1987335
radius_mean	2.0465578
concavity_worst	2.0262273
smoothness_worst	1.5309634
compactness_worst	1.3922155
symmetry_worst	1.0603018
radius_se	0.7813944
texture_se	0.5998619
compactness_mean	0.5998294
smoothness_mean	0.5911232
fractal_dimension_worst	0.5325797
fractal_dimension_se	0.5255427
perimeter_se	0.5088103
concavity_se	0.4972944
smoothness_se	0.4632677
compactness_se	0.4177373
symmetry_se	0.3354957
concave.points_se	0.3201936
symmetry_mean	0.2969199
fractal_dimension_mean	0.2536873

From this, we can see that **'perimeter_worst', 'concave.points_worst', 'area_worst', 'radius_worst', and 'concave.points_mean'** are the most important features.

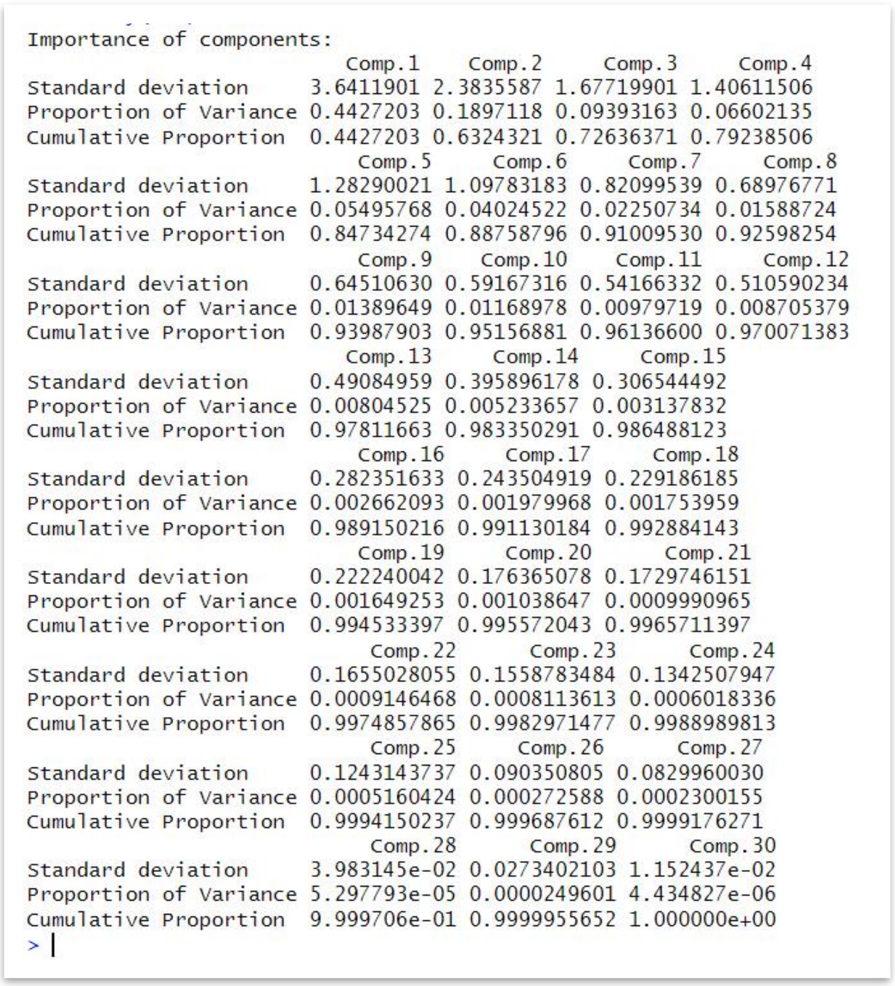
To visualise this further**:**



Principal Component Analysis (PCA) Overview:

PCA is a dimensionality reduction technique used to transform high-dimensional data into a lower-dimensional space while preserving most of the original variance.
It identifies the directions (principal components) that capture the maximum variance in the data

Importance of Components:
The standard deviation of each principal component represents the spread of data along that component.
Higher standard deviations indicate greater variability in the data along that component.



```
Importance of components:
                          Comp.1    Comp.2     Comp.3     Comp.4
Standard deviation     3.6411901 2.3835587 1.67719901 1.40611506
Proportion of Variance 0.4427203 0.1897118 0.09393163 0.06602135
Cumulative Proportion  0.4427203 0.6324321 0.72636371 0.79238506
                           Comp.5     Comp.6     Comp.7     Comp.8
Standard deviation     1.28290021 1.09783183 0.82099539 0.68976771
Proportion of Variance 0.05495768 0.04024522 0.02250734 0.01588724
Cumulative Proportion  0.84734274 0.88758796 0.91009530 0.92598254
                           Comp.9    Comp.10    Comp.11     Comp.12
Standard deviation     0.64510630 0.59167316 0.54166332 0.510590234
Proportion of Variance 0.01389649 0.01168978 0.00979719 0.008705379
Cumulative Proportion  0.93987903 0.95156881 0.96136600 0.970071383
                          Comp.13     Comp.14     Comp.15
Standard deviation     0.49084959 0.395896178 0.306544492
Proportion of Variance 0.00804525 0.005233657 0.003137832
Cumulative Proportion  0.97811663 0.983350291 0.986488123
                           Comp.16     Comp.17     Comp.18
Standard deviation     0.282351633 0.243504919 0.229186185
Proportion of Variance 0.002662093 0.001979968 0.001753959
Cumulative Proportion  0.989150216 0.991130184 0.992884143
                           Comp.19     Comp.20      Comp.21
Standard deviation     0.222240042 0.176365078 0.1729746151
Proportion of Variance 0.001649253 0.001038647 0.0009990965
Cumulative Proportion  0.994533397 0.995572043 0.9965711397
                            Comp.22      Comp.23      Comp.24
Standard deviation     0.1655028055 0.1558783484 0.1342507947
Proportion of Variance 0.0009146468 0.0008113613 0.0006018336
Cumulative Proportion  0.9974857865 0.9982971477 0.9988989813
                            Comp.25     Comp.26      Comp.27
Standard deviation     0.1243143737 0.090350805 0.0829960030
Proportion of Variance 0.0005160424 0.000272588 0.0002300155
Cumulative Proportion  0.9994150237 0.999687612 0.9999176271
                            Comp.28      Comp.29      Comp.30
Standard deviation     3.983145e-02 0.0273402103 1.152437e-02
Proportion of Variance 5.297793e-05 0.0000249601 4.434827e-06
Cumulative Proportion  9.999706e-01 0.9999955652 1.000000e+00
> |
```

Proportion of Variance:
The proportion of variance explained by each principal component indicates how much of the total variance in the dataset is captured by that component.
Components with higher proportions of variance are considered more important in explaining the data's variability.

Cumulative Proportion of Variance:

Cumulative proportion shows the total amount of variance explained by successive principal components.

It helps determine how many components are needed to capture a desired amount of variance.

Comp.1 explains the highest proportion of variance at 43.7%.

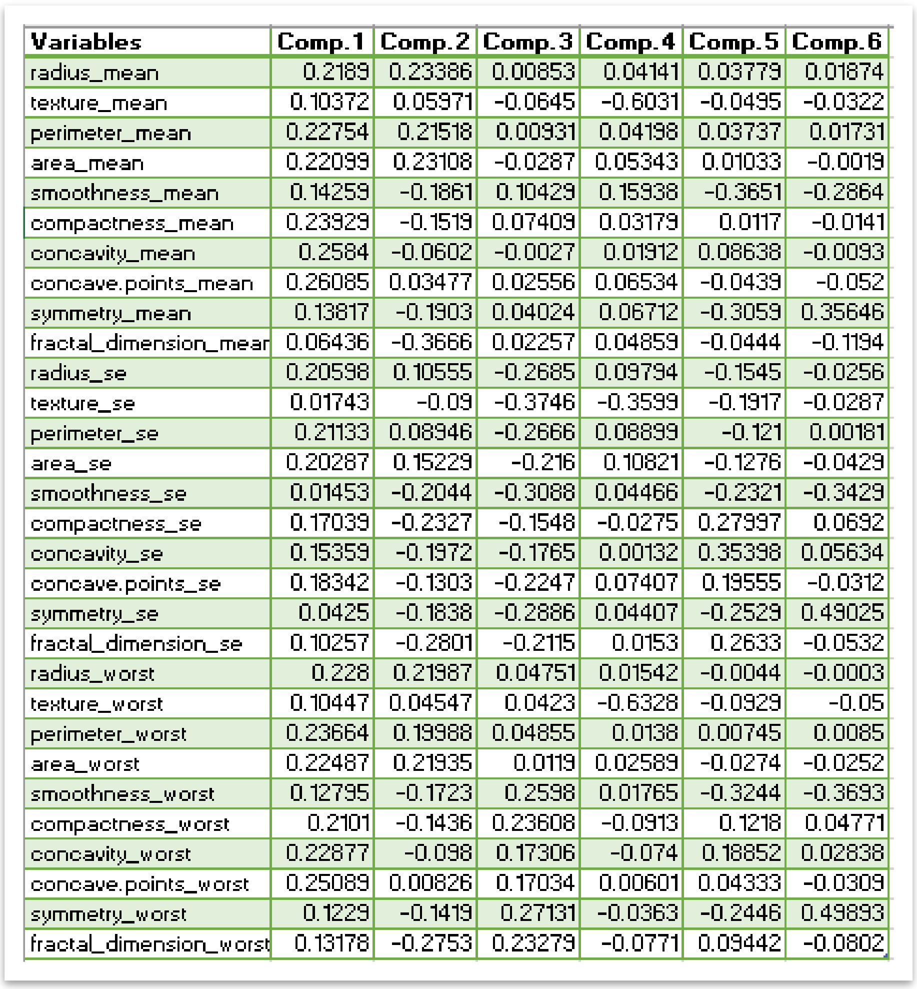
Comp.1 has a relatively high standard deviation of 3.56, indicating significant variability.

Loadings:

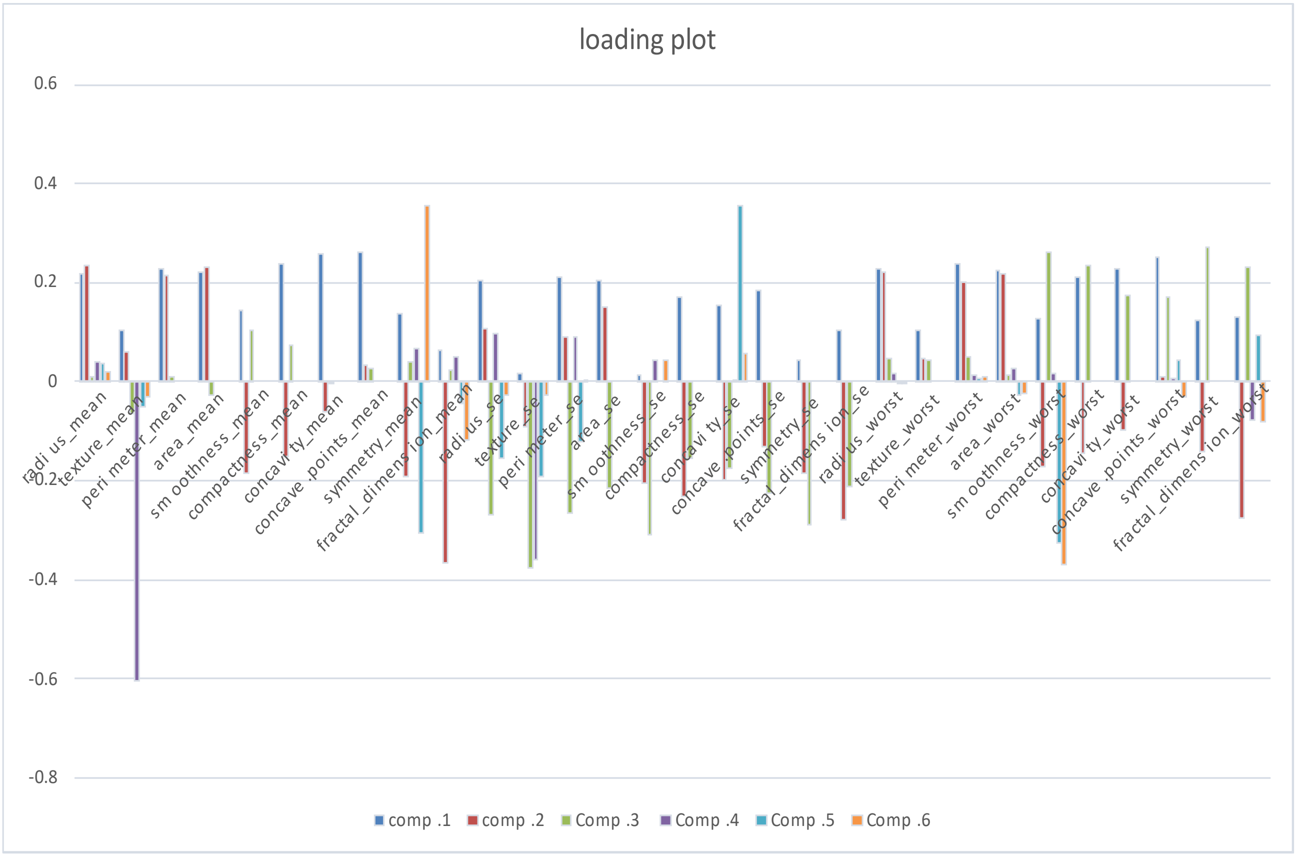
Loadings represent the correlation between the original variables and each principal component.

Positive loadings indicate a positive correlation, while negative loadings indicate a negative correlation.

Higher absolute loading values indicate a stronger relationship between the variable and the component.

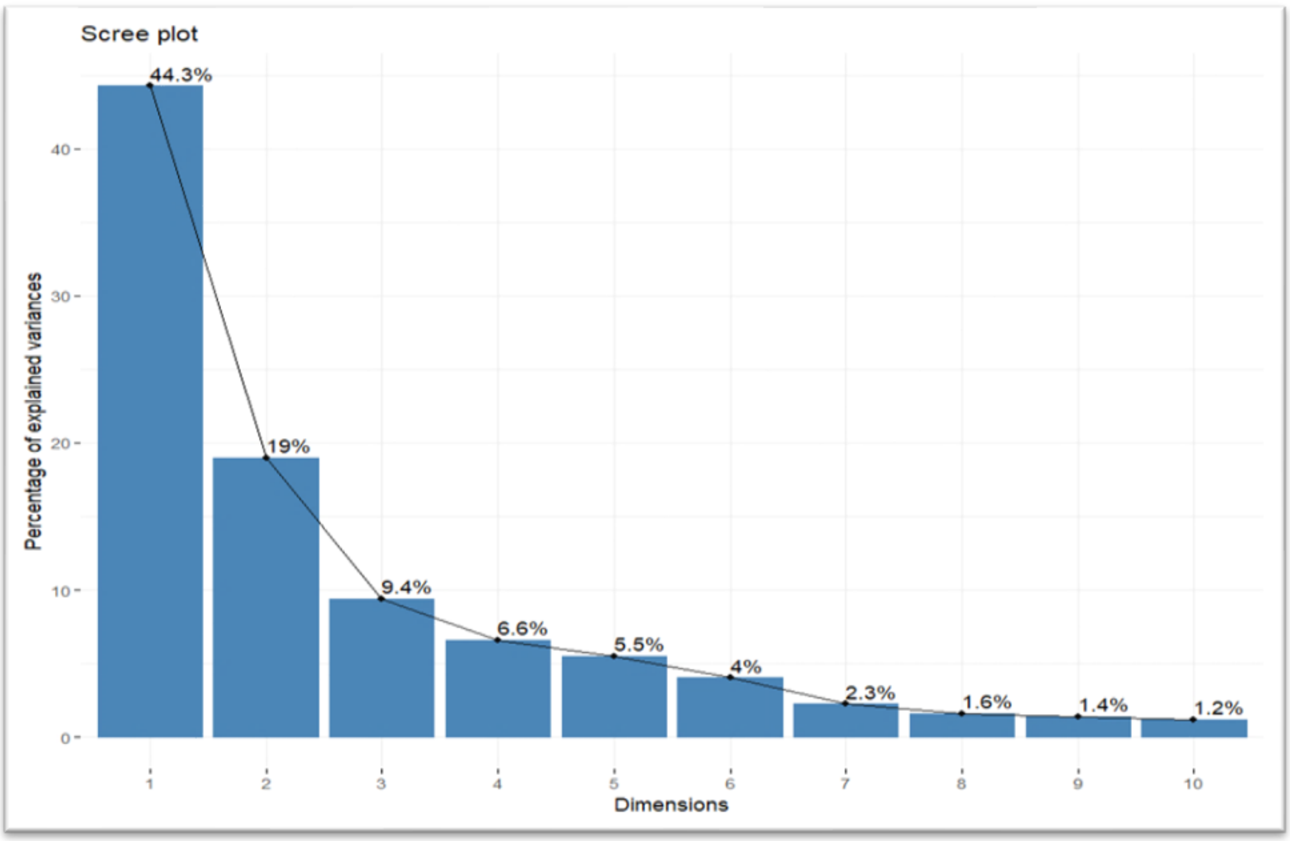


Variables	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6
radius_mean	0.2189	0.23386	0.00853	0.04141	0.03779	0.01874
texture_mean	0.10372	0.05971	-0.0645	-0.6031	-0.0495	-0.0322
perimeter_mean	0.22754	0.21518	0.00931	0.04198	0.03737	0.01731
area_mean	0.22099	0.23108	-0.0287	0.05343	0.01033	-0.0019
smoothness_mean	0.14259	-0.1861	0.10429	0.15938	-0.3651	-0.2864
compactness_mean	0.23929	-0.1519	0.07409	0.03179	0.0117	-0.0141
concavity_mean	0.2584	-0.0602	-0.0027	0.01912	0.08638	-0.0093
concave.points_mean	0.26085	0.03477	0.02556	0.06534	-0.0439	-0.052
symmetry_mean	0.13817	-0.1903	0.04024	0.06712	-0.3059	0.35646
fractal_dimension_mear	0.06436	-0.3666	0.02257	0.04859	-0.0444	-0.1194
radius_se	0.20598	0.10555	-0.2685	0.09794	-0.1545	-0.0256
texture_se	0.01743	-0.09	-0.3746	-0.3599	-0.1917	-0.0287
perimeter_se	0.21133	0.08946	-0.2666	0.08899	-0.121	0.00181
area_se	0.20287	0.15229	-0.216	0.10821	-0.1276	-0.0429
smoothness_se	0.01453	-0.2044	-0.3088	0.04466	-0.2321	-0.3429
compactness_se	0.17039	-0.2327	-0.1548	-0.0275	0.27997	0.0692
concavity_se	0.15359	-0.1972	-0.1765	0.00132	0.35398	0.05634
concave.points_se	0.18342	-0.1303	-0.2247	0.07407	0.19555	-0.0312
symmetry_se	0.0425	-0.1838	-0.2886	0.04407	-0.2529	0.49025
fractal_dimension_se	0.10257	-0.2801	-0.2115	0.0153	0.2633	-0.0532
radius_worst	0.228	0.21987	0.04751	0.01542	-0.0044	-0.0003
texture_worst	0.10447	0.04547	0.0423	-0.6328	-0.0929	-0.05
perimeter_worst	0.23664	0.19988	0.04855	0.0138	0.00745	0.0085
area_worst	0.22487	0.21935	0.0119	0.02589	-0.0274	-0.0252
smoothness_worst	0.12795	-0.1723	0.2598	0.01765	-0.3244	-0.3693
compactness_worst	0.2101	-0.1436	0.23608	-0.0913	0.1218	0.04771
concavity_worst	0.22877	-0.098	0.17306	-0.074	0.18852	0.02838
concave.points_worst	0.25089	0.00826	0.17034	0.00601	0.04333	-0.0309
symmetry_worst	0.1229	-0.1419	0.27131	-0.0363	-0.2446	0.49893
fractal_dimension_worst	0.13178	-0.2753	0.23279	-0.0771	0.09442	-0.0802



Selection of Principal Components:

The selection of principal components is crucial for reducing dimensionality while retaining as much variance as possible.
Based on the **screeplot** and cumulative proportion of variance using the "elbow method" wherein the number of components was determined using the plot such that it is the value preceding the value where the elbow was formed, the top 6 principal components were chosen for further analysis.
These components provide a compact representation of the original dataset while preserving most of its variability.



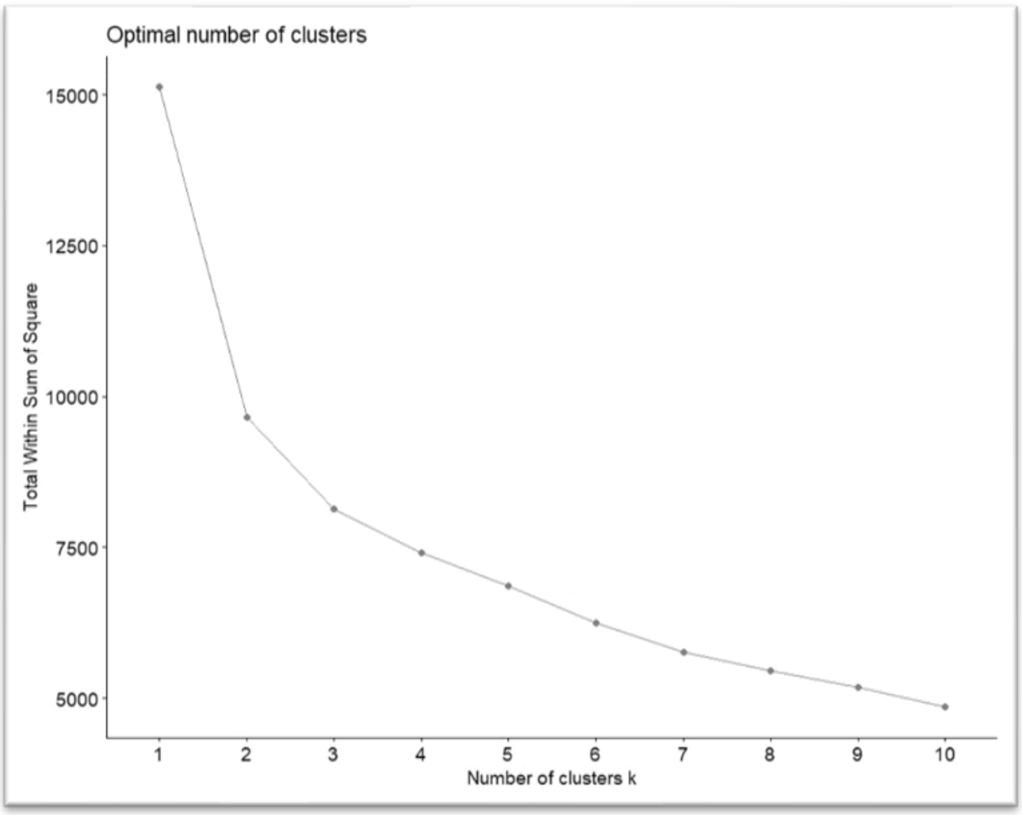
The top 6 components (Comp.1 to Comp.6) cumulatively explain approximately 89% of the total variance, suggesting that these components capture most of the data's variability.

Transformation and Further Analysis:
The original dataset was transformed using the selected principal components (Comp.1 to Comp.6), resulting in a lower-dimensional representation.
This transformed dataset can be used for various downstream tasks such as clustering, classification, or visualization.

Conclusion:
The PCA results offer valuable insights into the underlying structure and variability of the dataset. By selecting the most informative principal components, we can effectively reduce dimensionality without significant loss of information, enabling more efficient analysis and interpretation of the data.

K-means Clustering

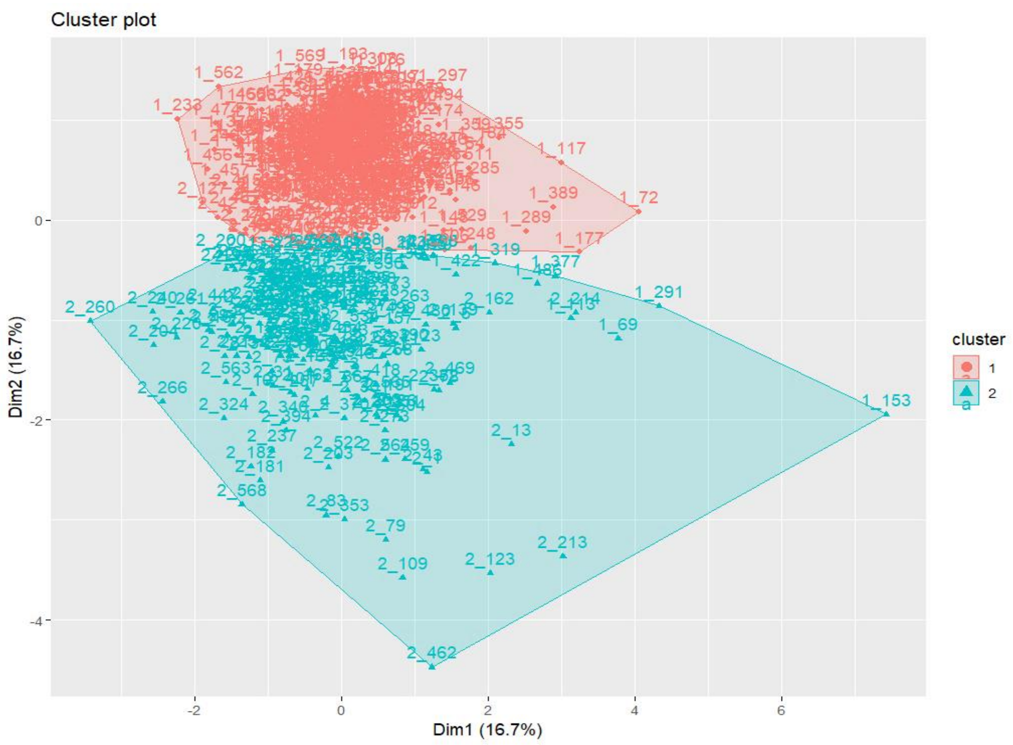
Determination of the Optimal number of clusters
The graph of the number of clusters v/s Total within the sum of squares was plotted to find the number of clusters to be used for k-means clustering
The optimal number of clusters was determined using the "elbow method" wherein 'k' was determined using the plot such that it is the value preceding the value where the elbow was formed (k=3).
Thus, the optimal number was found to be k=2



Results
The k-means algorithm identified two clusters:
Cluster 1: 380 data points
Cluster 2: 189 data points
The distribution of actual diagnosis labels among the clusters is as follows:

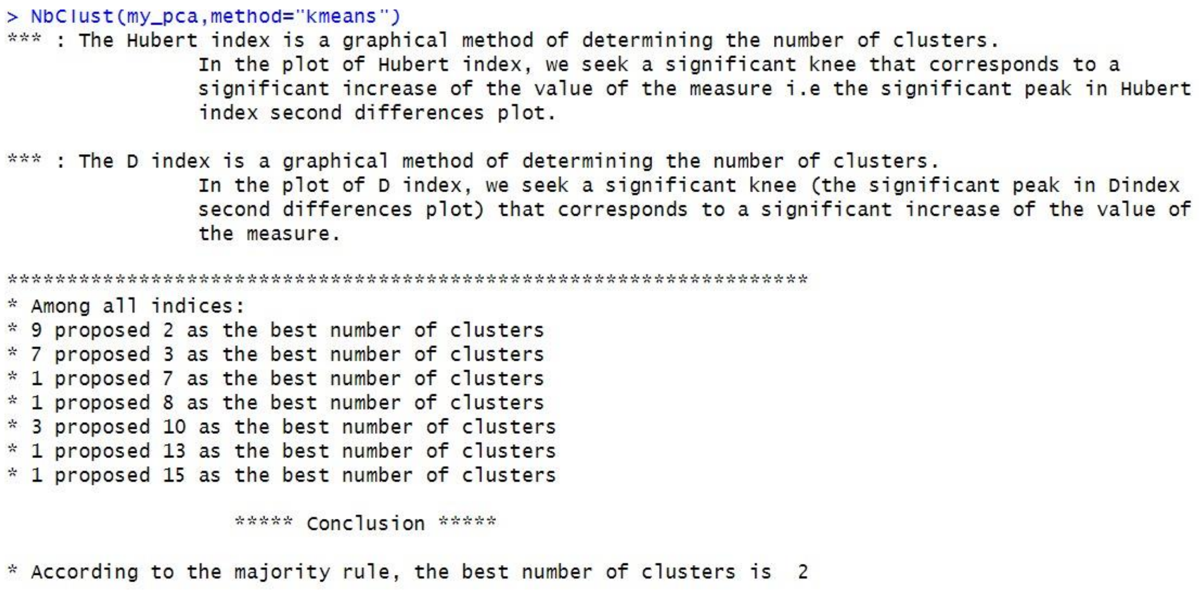
Diagnosis	Cluster 1	Cluster 2
Benign	343	14
Malignant	37	175

Cluster Interpretation:
Cluster 1 appears to predominantly contain benign cases, as evidenced by the high number of benign diagnosis labels. Conversely, Cluster 2 seems to have a higher proportion of malignant cases



Verification:

Using the built-in library NbClust on R, we can see that the **maximum number of indices suggests the optimal number of clusters to be 2**, in turn verifying our evaluations.



```
> NbClust(my_pca,method="kmeans")
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in Hubert
                index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the significant peak in Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure.

*******************************************************************
* Among all indices:
* 9 proposed 2 as the best number of clusters
* 7 proposed 3 as the best number of clusters
* 1 proposed 7 as the best number of clusters
* 1 proposed 8 as the best number of clusters
* 3 proposed 10 as the best number of clusters
* 1 proposed 13 as the best number of clusters
* 1 proposed 15 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  2
```

Evaluations and conclusion:

The clustering model seems to have effectively separated the data into two distinct groups, aligning well with the actual diagnosis labels

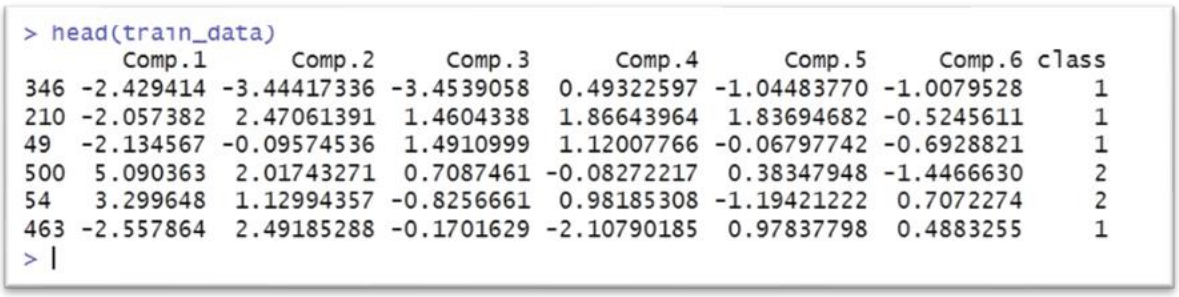
Predictive modelling:

In this section, we discuss the predictive modelling aspect of our project, where we aim to develop machine learning models to predict the outcome variable based on the input features.

Data Splitting:
To facilitate the development and evaluation of predictive models, we divided our dataset into two subsets: a training set and a testing set. The training set comprises 70% of the original data, while the testing set contains the remaining 30%.

Model Training:
The larger portion of the data (70%) allocated to the training set allows the model to learn patterns and relationships from a substantial amount of data. This aids in building robust and well-generalized models.
Model Evaluation:
The testing set, representing 30% of the data, serves as an independent dataset for evaluating the performance of the trained model. By withholding a portion of the data during training, we can assess how well the model performs on unseen data, thus estimating its real-world performance.
Reducing Overfitting:
Allocating a larger portion of the data to the training set helps mitigate the risk of overfitting, where the model learns to memorize the training data rather than capturing underlying patterns. The testing set acts as a safeguard against overfitting by providing a separate dataset for validation.
Statistical Significance:
A 70:30 split strikes a balance between having sufficient data for model training and ensuring statistically significant evaluation results on the testing set. It is a commonly used ratio in machine learning practice and has been shown to yield reliable model performance estimates.

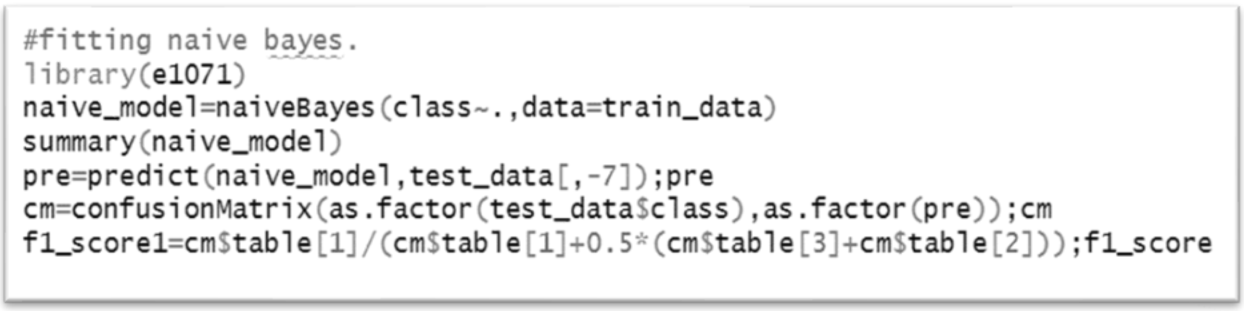
Following are a few values from the training dataset



```
> head(train_data)
       Comp.1      Comp.2     Comp.3      Comp.4      Comp.5     Comp.6 class
346 -2.429414 -3.44417336 -3.4539058  0.49322597 -1.04483770 -1.0079528     1
210 -2.057382  2.47061391  1.4604338  1.86643964  1.83694682 -0.5245611     1
49  -2.134567 -0.09574536  1.4910999  1.12007766 -0.06797742 -0.6928821     1
500  5.090363  2.01743271  0.7087461 -0.08272217  0.38347948 -1.4466630     2
54   3.299648  1.12994357 -0.8256661  0.98185308 -1.19421222  0.7072274     2
463 -2.557864  2.49185288 -0.1701629 -2.10790185  0.97837798  0.4883255     1
> |
```

Here, we have applied three different categorization models for the diagnosis of breast cancer: Naïve Bayes, k-nearest neighbours (KNN), and Logistic regression (LR).

Naïve Bayes



```
#fitting naive bayes.
library(e1071)
naive_model=naiveBayes(class~.,data=train_data)
summary(naive_model)
pre=predict(naive_model,test_data[,-7]);pre
cm=confusionMatrix(as.factor(test_data$class),as.factor(pre));cm
f1_score1=cm$table[1]/(cm$table[1]+0.5*(cm$table[3]+cm$table[2]));f1_score
```

Confusion matrix:

	Reference	
Prediction	1	2
1	101	6
2	9	55

Accuracy:
The Naive Bayes model achieved an accuracy of approximately 91.23% on the testing data, indicating its overall effectiveness in classification.

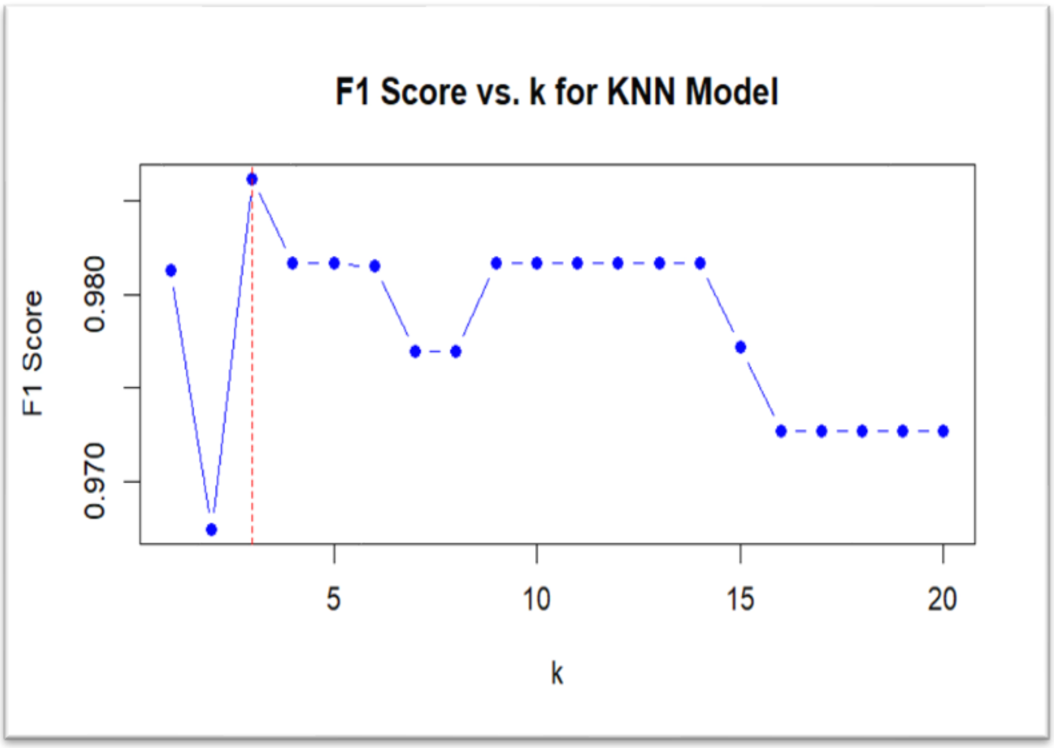
Sensitivity and Specificity:
The model demonstrated good sensitivity (91.82%) and specificity (90.16%), suggesting its ability to correctly identify both positive and negative instances.

F1 Score:
The F1 score, a measure of a model's accuracy, precision, and recall, was calculated to be approximately 93.08756%, indicating a good balance between precision and recall.

K-Nearest Neighbour Classifier

K- Tuning

To find the optimal value of k we have used a trial-and-error method using a ‘for-loop’. Here, we tested different values of k ranging from 1 to 20 by calculating their F1 scores and choosing the value of k with the highest F1 score as the optimal value.



This plot is created to visualize how the F1 score changes with varying values of K. This plot helps in identifying the optimal value of K that maximizes the model's performance.

Results:
The maximum value of F1-score is observed for k=3
Thus, it is the optimal value of the k-nearest neighbour.

	Reference	
Prediction	1	2
1	107	3
2	0	61

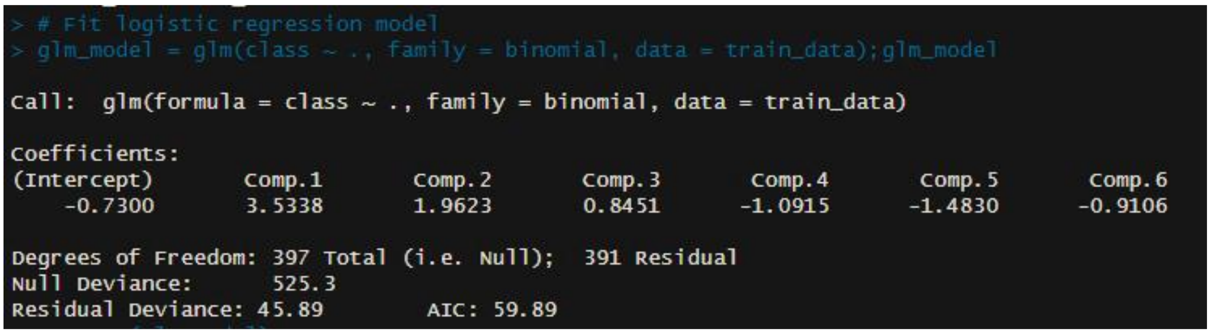
Accuracy:
The Naive Bayes model achieved an accuracy of approximately 98.25% on the testing data, indicating its overall effectiveness in classification.

Sensitivity and Specificity:
The model demonstrated good sensitivity (100%) and specificity (95.31%), suggesting its ability to correctly identify both positive and negative instances.

F1 Score:
The F1 score, a measure of a model's accuracy, precision, and recall, was calculated to be approximately 98.61%, indicating a good balance between precision and recall.

Logistic Regression

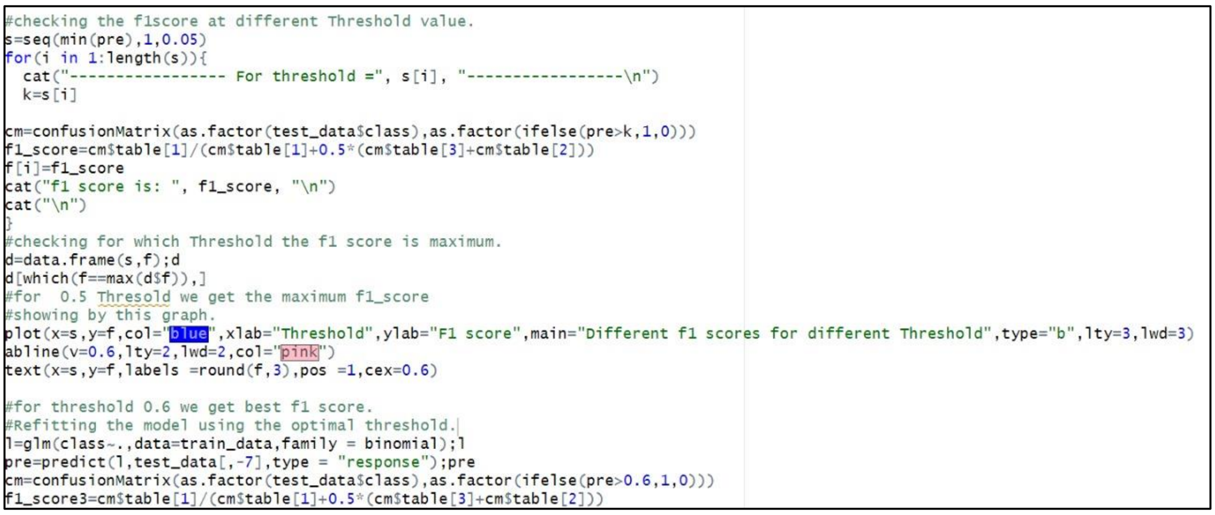
Here, we fit a logistic regression model by converting our classes to binary, such that, Benign:0, Malignant: 1
We fit the model on the training dataset and make the predictions based on the testing dataset.



```
> # Fit logistic regression model
> glm_model = glm(class ~ ., family = binomial, data = train_data);glm_model

Call:  glm(formula = class ~ ., family = binomial, data = train_data)

Coefficients:
(Intercept)       Comp.1       Comp.2       Comp.3       Comp.4       Comp.5       Comp.6
    -0.7300       3.5338       1.9623       0.8451      -1.0915      -1.4830      -0.9106

Degrees of Freedom: 397 Total (i.e. Null);  391 Residual
Null Deviance:      525.3
Residual Deviance: 45.89        AIC: 59.89
```

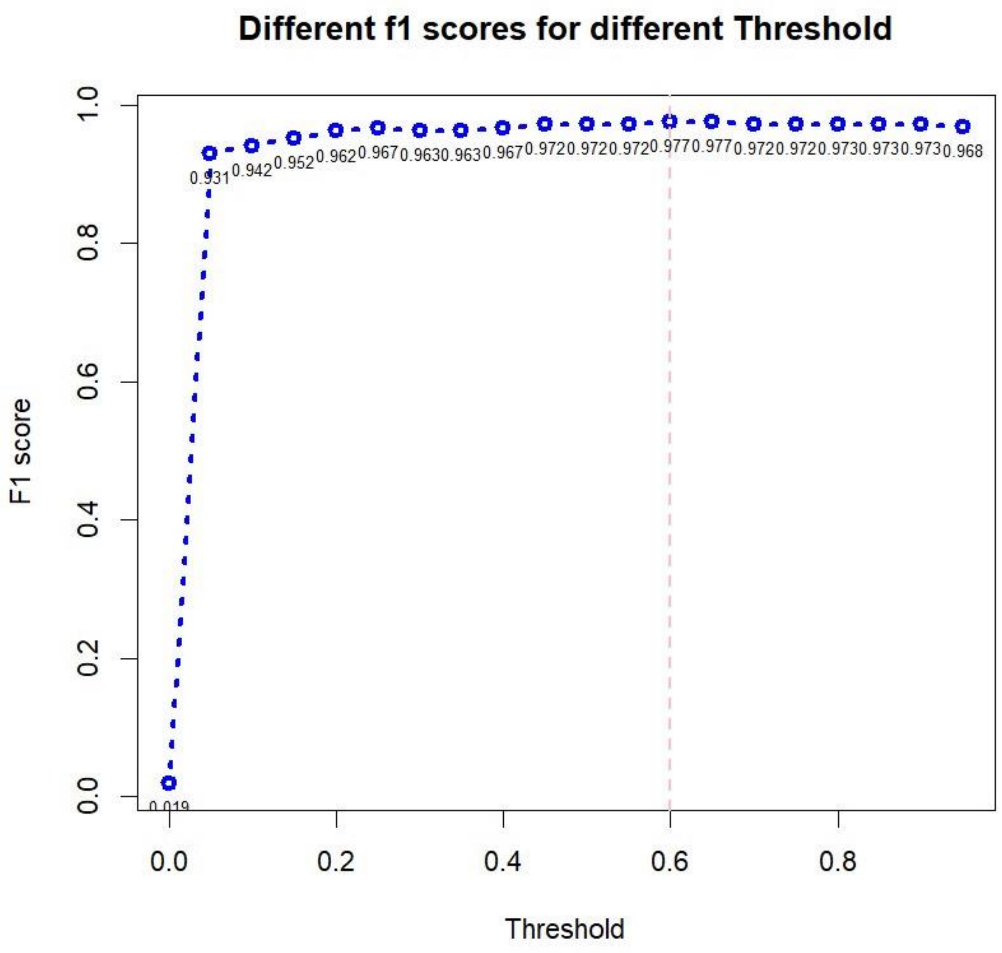
Here,

We evaluate the performance of the model by varying the threshold for classifying predictions.
The following loop calculates the f1 score at different threshold values, which balances precision and recall.
The process involves applying each threshold to the model predictions, creating a confusion matrix to compare predictions with actual labels, and computing the f1 score.
The f1 scores are stored for analysis to determine the optimal threshold for the model.
Then the model was refitted using the optimal threshold to make further predictions.



```
#checking the f1score at different Threshold value.
s=seq(min(pre),1,0.05)
for(i in 1:length(s)){
  cat("----------------- For threshold =", s[i], "-----------------\n")
  k=s[i]

cm=confusionMatrix(as.factor(test_data$class),as.factor(ifelse(pre>k,1,0)))
f1_score=cm$table[1]/(cm$table[1]+0.5*(cm$table[3]+cm$table[2]))
f[i]=f1_score
cat("f1 score is: ", f1_score, "\n")
cat("\n")
}
#checking for which Threshold the f1 score is maximum.
d=data.frame(s,f);d
d[which(f==max(d$f)),]
#for  0.5 Thresold we get the maximum f1_score
#showing by this graph.
plot(x=s,y=f,col="blue",xlab="Threshold",ylab="F1 score",main="Different f1 scores for different Threshold",type="b",lty=3,lwd=3)
abline(v=0.6,lty=2,lwd=2,col="pink")
text(x=s,y=f,labels =round(f,3),pos =1,cex=0.6)

#for threshold 0.6 we get best f1 score.
#Refitting the model using the optimal threshold.
l=glm(class~.,data=train_data,family = binomial);l
pre=predict(l,test_data[,-7],type = "response");pre
cm=confusionMatrix(as.factor(test_data$class),as.factor(ifelse(pre>0.6,1,0)))
f1_score3=cm$table[1]/(cm$table[1]+0.5*(cm$table[3]+cm$table[2]))
```

The plot of F1 score vs Threshold values:



Here we get the optimal threshold as 0.6 with the maximum f1 score of 97.8%.

Results and Interpretations:

Coefficients:

The intercept term (Intercept) is -0.7300. This represents the logit of the baseline outcome when all predictor variables are zero. The coefficients for the predictor variables (Comp.1, Comp.2, Comp.3, Comp.4, Comp.5, and Comp.6) represent the change in the logit of the outcome for a one-unit change in the corresponding predictor, holding other predictors constant. A one-unit increase in Comp.1 corresponds to an increase in the logit of the outcome by 3.5338 units, all else being equal. The p-values associated with each coefficient indicate the significance of the relationship between each predictor and the outcome. Lower p-values (< 0.05) suggest a significant relationship.

Deviance Residuals:
The minimum and maximum deviance residuals are -1.73985 and 2.78289, respectively. Ideally, residuals should be close to zero, indicating a good fit.

Model Fit:
The residual Model Fit: The null deviance represents the deviance of the model with no predictors, while the residual deviance represents the deviance of the model with predictors. The null deviance of 525.311 on 397 degrees of freedom suggests that the model without predictors doesn't fit well. The residual
deviance of 45.892 on 391 degrees of freedom indicates that the model with predictors fits significantly better than the null model.
The AIC (Akaike Information Criterion) value of 59.892 is a measure of the relative quality of the model. Lower AIC values indicate a better fit, suggesting that the current model is relatively good
The residual deviance of 45.892 on 391 degrees of freedom indicates that the model with predictors fits significantly better than the null model.

Significance of Predictors:
Comp.1, Comp.2, Comp.3, Comp.4, Comp.5, and Comp.6 have significant coefficients ($p < 0.05$), indicating that they are likely important predictors of the outcome.
Comp.1, Comp.2, Comp.3, and Comp.4 have particularly low p-values (< 0.001), suggesting stronger evidence of their significance.

Conclusion:
In conclusion, the logistic regression model indicates that the predictors (Comp.1 to Comp.6) are significantly associated with the outcome variable. The model fits the data well, as evidenced by the low residual deviance and AIC value. The interpretation of coefficients suggests the direction and strength of the relationship between predictors and the outcome

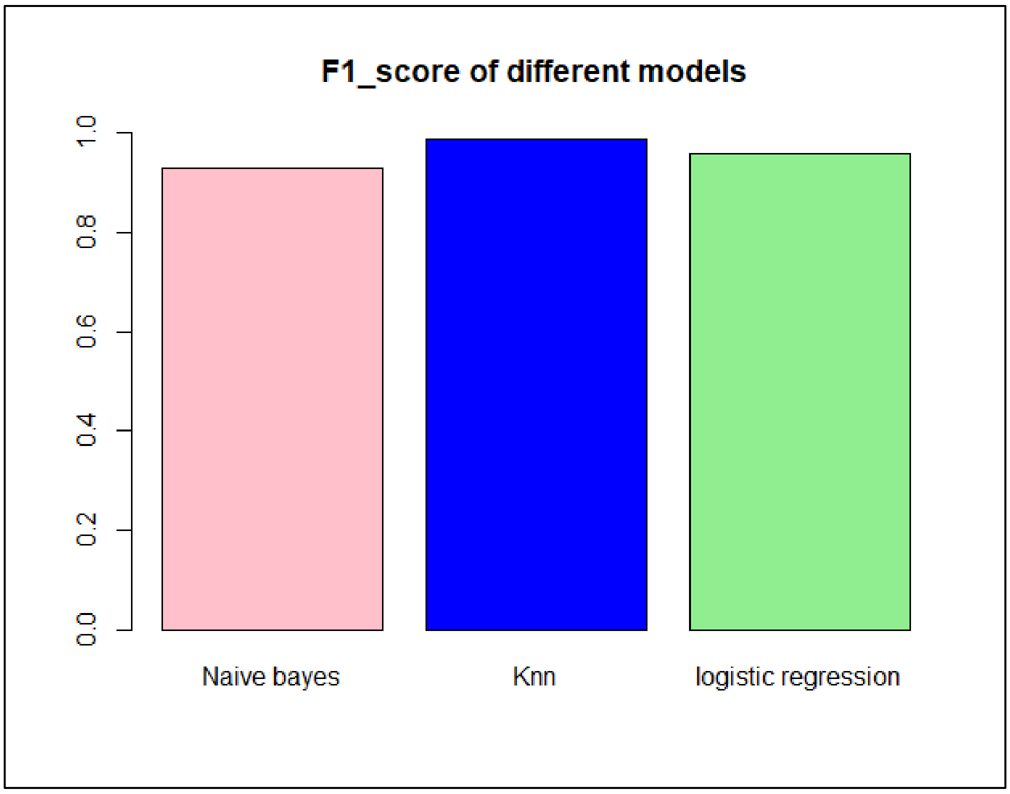
Comparison of fitted models:

For comparison, we have preferred the F1 score over the accuracy, precision, and recall because:

The F1 score provides a balanced measure of a model's performance by considering both precision and recall. Unlike accuracy, which can be misleading when classes are imbalanced, the F1 score takes into account both false positives (precision) and false negatives (recall), providing a more comprehensive assessment of a classifier's effectiveness.

The considered dataset is imbalanced, meaning the ‘Benign’ class is significantly more prevalent than others, accuracy alone can be misleading, as a model that predicts the majority class for all instances can achieve high accuracy while performing poorly on the minority class. F1 score, on the other hand, considers the trade-off between precision and recall, making it a more suitable metric for imbalanced datasets.

Different models may have different optimal decision thresholds for classification. By using the F1 score, which considers precision and recall at various thresholds, we can better understand the model's performance across different decision boundaries.



By comparison of the F1 scores of all three fitted models, it can be seen that KNN is the best-fitted model for our dataset.

Project Results:

1. Revealing Predictive Factors:
Through thorough analysis of the dataset, we identified 30 potential predictive factors for early-stage breast cancer. After conducting a feature importance analysis, the variable **'perimeter_worst' ,'concave.points_worst', 'area_worst', 'radius_worst', and 'concave.points_mean'** was found to be the most influential predictor of breast cancer using random forest.

2. Comprehensive Comparison and Analysis:
We employed machine learning techniques to compare the performance of three algorithms: K-Nearest Neighbours (KNN), Naive Bayes, and Logistic Regression. After evaluating the models using F1 scores, KNN was identified as the most accurate predictor of breast cancer among the three algorithms.

3. Evaluation of Machine Learning Algorithms:
We assessed the performance of each machine learning algorithm using various evaluation metrics, including accuracy, precision, recall, and F1 score. Based on the evaluation results, the model with the highest F1 score,98.61% was selected as the preferred model for breast cancer prediction.

4. Clustering Analysis:
Clustering analysis was conducted to uncover distinct groups of breast cancer patients within the dataset.
By evaluating the optimal number of clusters, 2 clusters were identified, effectively separating breast cancer classes and providing valuable insights into the data.

Conclusion:

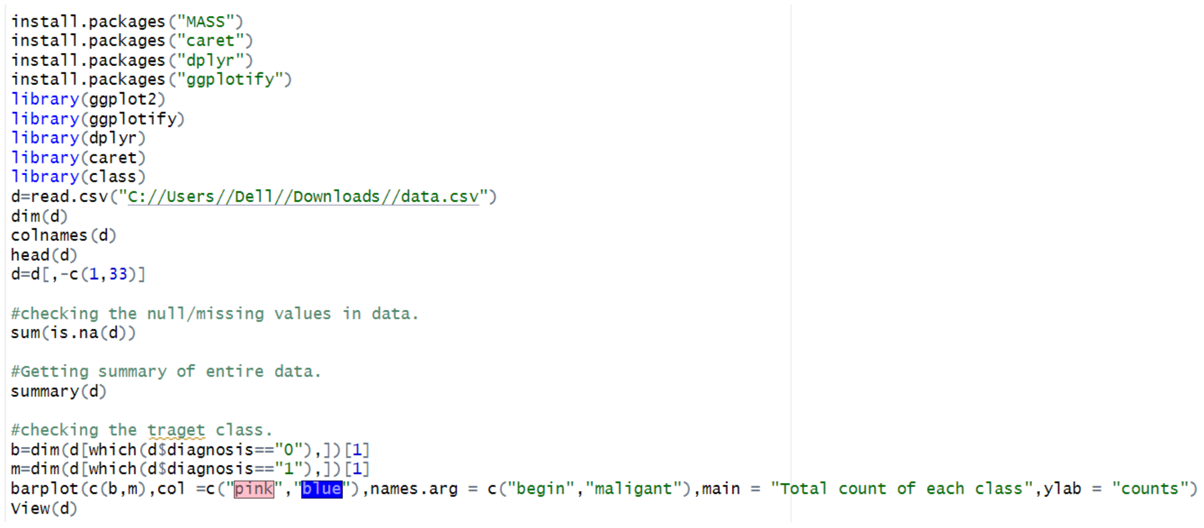
Through a systematic approach encompassing feature analysis, model comparison, and clustering analysis, we successfully achieved the objectives outlined for the project. The utilization of machine learning algorithms, identification of significant features, and validation of clustering results have collectively contributed to the establishment of an effective model for early-stage breast cancer prediction.

BIBLIOGRAPHY

- University of California Irvine (UCI) Machine Learning Repository:
  www.archive.ics.uci.edu
- Google Scholar
  www.scholar.google.com
- Google
  www.google.co.in
- Youtube
  www.youtube.com
- Softwares used
  R software
  Microsoft Excel
  Python

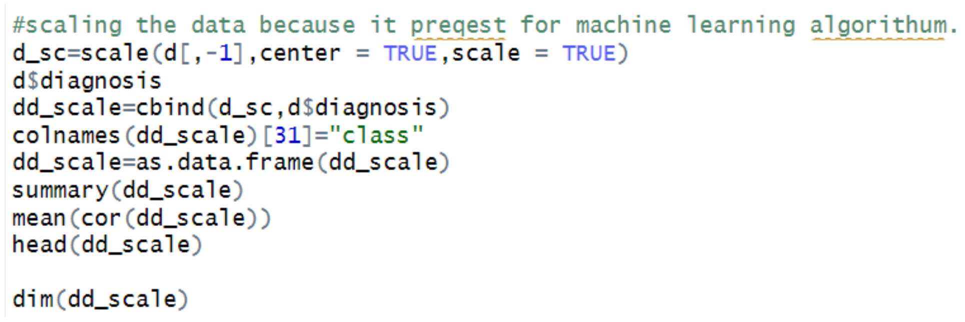
APPENDIX

Data and data summary



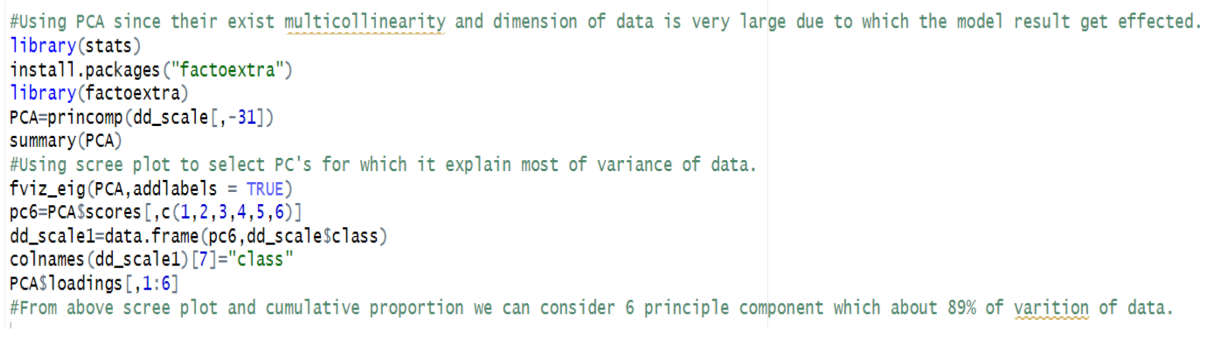
```
install.packages("MASS")
install.packages("caret")
install.packages("dplyr")
install.packages("ggplotify")
library(ggplot2)
library(ggplotify)
library(dplyr)
library(caret)
library(class)
d=read.csv("C://Users//Dell//Downloads//data.csv")
dim(d)
colnames(d)
head(d)
d=d[,-c(1,33)]

#checking the null/missing values in data.
sum(is.na(d))

#Getting summary of entire data.
summary(d)

#checking the traget class.
b=dim(d[which(d$diagnosis=="0"),])[1]
m=dim(d[which(d$diagnosis=="1"),])[1]
barplot(c(b,m),col =c("pink","blue"),names.arg = c("begin","maligant"),main = "Total count of each class",ylab = "counts")
View(d)
```

Scaling of the data



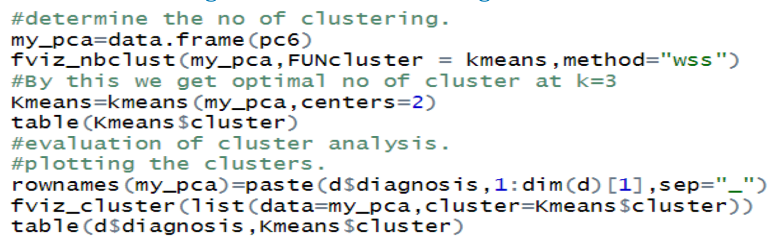
```
#scaling the data because it preqest for machine learning algorithum.
d_sc=scale(d[,-1],center = TRUE,scale = TRUE)
d$diagnosis
dd_scale=cbind(d_sc,d$diagnosis)
colnames(dd_scale)[31]="class"
dd_scale=as.data.frame(dd_scale)
summary(dd_scale)
mean(cor(dd_scale))
head(dd_scale)

dim(dd_scale)
```

PCA



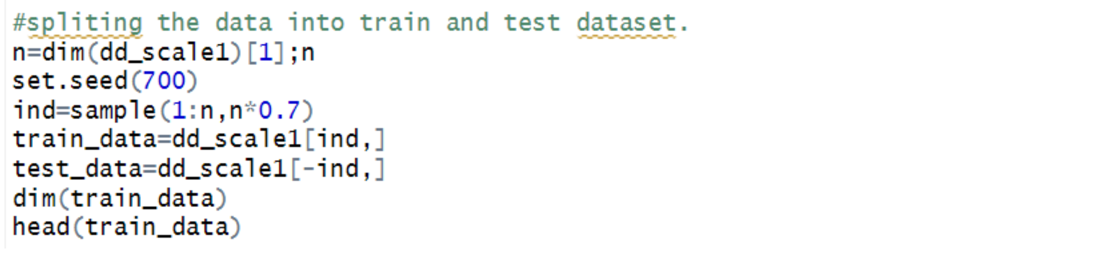
```
#Using PCA since their exist multicollinearity and dimension of data is very large due to which the model result get effected.
library(stats)
install.packages("factoextra")
library(factoextra)
PCA=princomp(dd_scale[,-31])
summary(PCA)
#Using scree plot to select PC's for which it explain most of variance of data.
fviz_eig(PCA,addlabels = TRUE)
pc6=PCA$scores[,c(1,2,3,4,5,6)]
dd_scale1=data.frame(pc6,dd_scale$class)
colnames(dd_scale1)[7]="class"
PCA$loadings[,1:6]
#From above scree plot and cumulative proportion we can consider 6 principle component which about 89% of varition of data.
```

Clustering and K-means clustering



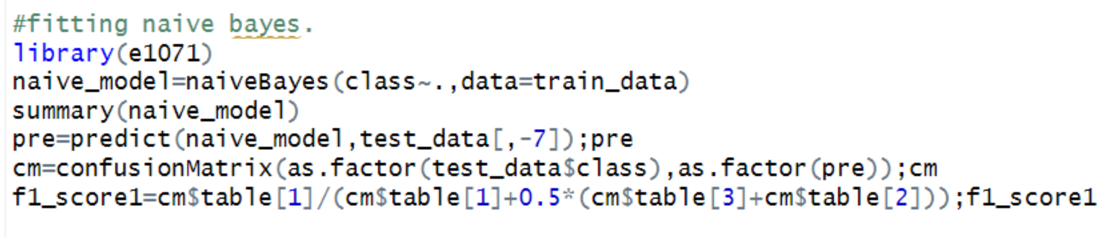
```
#determine the no of clustering.
my_pca=data.frame(pc6)
fviz_nbclust(my_pca,FUNcluster = kmeans,method="wss")
#By this we get optimal no of cluster at k=3
Kmeans=kmeans(my_pca,centers=2)
table(Kmeans$cluster)
#evaluation of cluster analysis.
#plotting the clusters.
rownames(my_pca)=paste(d$diagnosis,1:dim(d)[1],sep="_")
fviz_cluster(list(data=my_pca,cluster=Kmeans$cluster))
table(d$diagnosis,Kmeans$cluster)
```

- Data splitting (70-30)



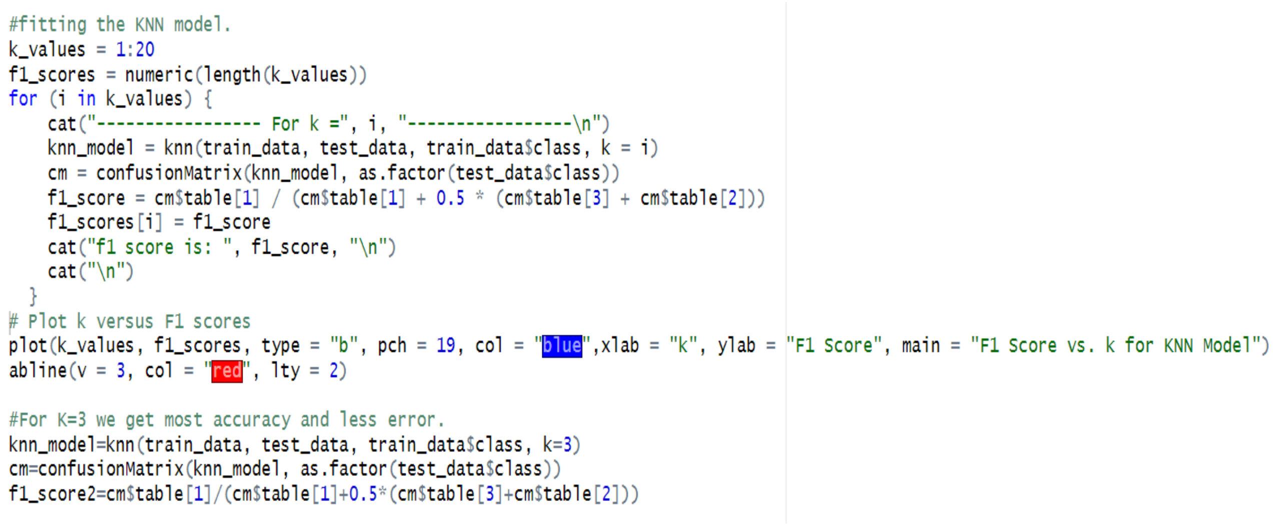
```
#spliting the data into train and test dataset.
n=dim(dd_scale1)[1];n
set.seed(700)
ind=sample(1:n,n*0.7)
train_data=dd_scale1[ind,]
test_data=dd_scale1[-ind,]
dim(train_data)
head(train_data)
```

- Naïve Bayes



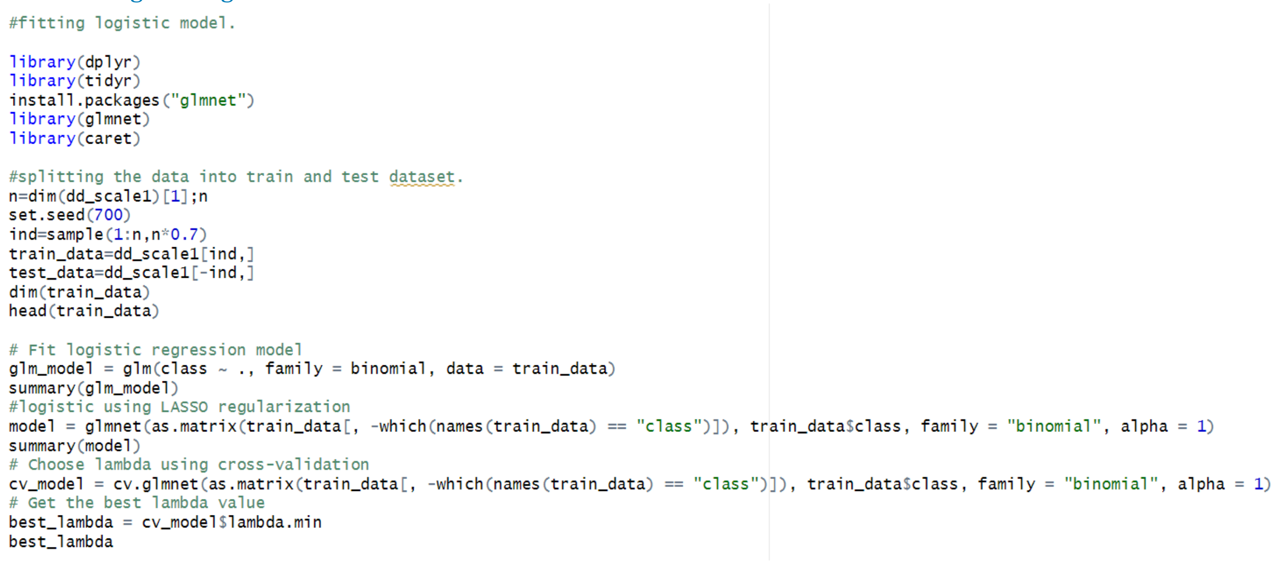
```
#fitting naive bayes.
library(e1071)
naive_model=naiveBayes(class~.,data=train_data)
summary(naive_model)
pre=predict(naive_model,test_data[,-7]);pre
cm=confusionMatrix(as.factor(test_data$class),as.factor(pre));cm
f1_score1=cm$table[1]/(cm$table[1]+0.5*(cm$table[3]+cm$table[2]));f1_score1
```

- KNN Classifiers



```
#fitting the KNN model.
k_values = 1:20
f1_scores = numeric(length(k_values))
for (i in k_values) {
    cat("---------------- For k =", i, "----------------\n")
    knn_model = knn(train_data, test_data, train_data$class, k = i)
    cm = confusionMatrix(knn_model, as.factor(test_data$class))
    f1_score = cm$table[1] / (cm$table[1] + 0.5 * (cm$table[3] + cm$table[2]))
    f1_scores[i] = f1_score
    cat("f1 score is: ", f1_score, "\n")
    cat("\n")
  }
# Plot k versus F1 scores
plot(k_values, f1_scores, type = "b", pch = 19, col = "blue",xlab = "k", ylab = "F1 Score", main = "F1 Score vs. k for KNN Model")
abline(v = 3, col = "red", lty = 2)

#For K=3 we get most accuracy and less error.
knn_model=knn(train_data, test_data, train_data$class, k=3)
cm=confusionMatrix(knn_model, as.factor(test_data$class))
f1_score2=cm$table[1]/(cm$table[1]+0.5*(cm$table[3]+cm$table[2]))
```

- Logistic regression



```
#fitting logistic model.

library(dplyr)
library(tidyr)
install.packages("glmnet")
library(glmnet)
library(caret)

#splitting the data into train and test dataset.
n=dim(dd_scale1)[1];n
set.seed(700)
ind=sample(1:n,n*0.7)
train_data=dd_scale1[ind,]
test_data=dd_scale1[-ind,]
dim(train_data)
head(train_data)

# Fit logistic regression model
glm_model = glm(class ~ ., family = binomial, data = train_data)
summary(glm_model)
#logistic using LASSO regularization
model = glmnet(as.matrix(train_data[, -which(names(train_data) == "class")]), train_data$class, family = "binomial", alpha = 1)
summary(model)
# Choose lambda using cross-validation
cv_model = cv.glmnet(as.matrix(train_data[, -which(names(train_data) == "class")]), train_data$class, family = "binomial", alpha = 1)
# Get the best lambda value
best_lambda = cv_model$lambda.min
best_lambda
```

```
# Refit the model with the best lambda
model_best = glmnet(as.matrix(train_data[, -which(names(train_data) == "class")]), train_data$class, family = "binomial", alpha = 1, lambda = best_lambda)
summary(model_best)

# Predict on test data using the LASSO model
pre = predict(model_best, newx = as.matrix(test_data[, -which(names(test_data) == "class")]), type = "response")
pre
```

```
# Convert predicted probabilities to binary predictions (0 or 1)
pre1 = ifelse(pre > 0.5, 1, 0)
pre1
accuracy = mean(pre1 == test_data$class)
cat("Accuracy with LASSO regularization:", accuracy , "\n")

# confusion matrix
conf_matrix = table(Actual = test_data$class, Predicted = pre1)
conf_matrix
# F1 score
precision = conf_matrix[2, 2] / sum(conf_matrix[, 2])
recall = conf_matrix[2, 2] / sum(conf_matrix[2, ])
f1_score3 = 2 * (precision * recall) / (precision + recall)

cat("F1 Score with LASSO regularization:", f1_score3, "\n")
```

F1 Scores for different threshold values:

```
# Finding F1 scores for different threshold values

s=seq(0, 1, 0.05)
f=numeric(length = length(s))

for (i in 2:(length(s) - 1)) {
  cat("------------------ For threshold =", s[i], "------------------\n")

  k = s[i]

  pred_factor = factor(ifelse(pre > k, 1, 0), levels = c("0", "1"))
  cm = confusionMatrix(as.factor(test_data$class), pred_factor)
  precision = cm$byClass["Pos Pred Value"]
  recall = cm$byClass["Sensitivity"]

  if (!is.na(precision) && !is.na(recall)) {
    f1_score = 2 * (precision * recall) / (precision + recall)

    f[i] = f1_score

    cat("F1 score is:", f1_score, "\n\n")
  } else {
    cat("F1 score is: NA\n\n")
  }
}

# Find the threshold value with maximum F1 score
max_f1_threshold = s[which.max(f)]
max_f1_score = max(f)

cat("Threshold value with maximum F1 score:", max_f1_threshold, "\n")
cat("Maximum F1 score:", max_f1_score, "\n")
```

Plot of F1-Scores for different thresholds:

```
#Plotting F1 scores for different threshold values
plot(x = s, y = f, col = ifelse(s == max_f1_threshold, "red", "blue"),
     xlab = "Threshold", ylab = "F1 Score",
     main = "F1 Scores for Different Thresholds", type = "b", lty = 3, lwd = 3)

abline(v = max_f1_threshold, lty = 2, lwd = 2, col = "pink")
text(x = s, y = f, labels = round(f, 3), pos = 1, cex = 0.6)
legend("bottomright", legend = c("Maximum F1 Score", "F1 Scores"), col = c("pink", "blue"), lty = c(2, 1),
       lwd = c(2, 3), cex = 0.8, bg = "white")

#confusion matrix
conf_matrix1 = confusionMatrix(factor(pre1), factor(test_data$class))
conf_matrix1
```

- Comparison of the Predictive Models

```
#confusion:
d=data.frame(Models=c("Naive bayes","Knn","logistic regression"),f1_score=c(f1_score1,f1_score2,f1_score3));d
barplot(c(f1_score1,f1_score2,f1_score3),names.arg = c("Naive bayes","Knn","logistic regression"),col=c("pink","blue","light green" ),ylim=c(0,1),main = "F1_score of different models")
```