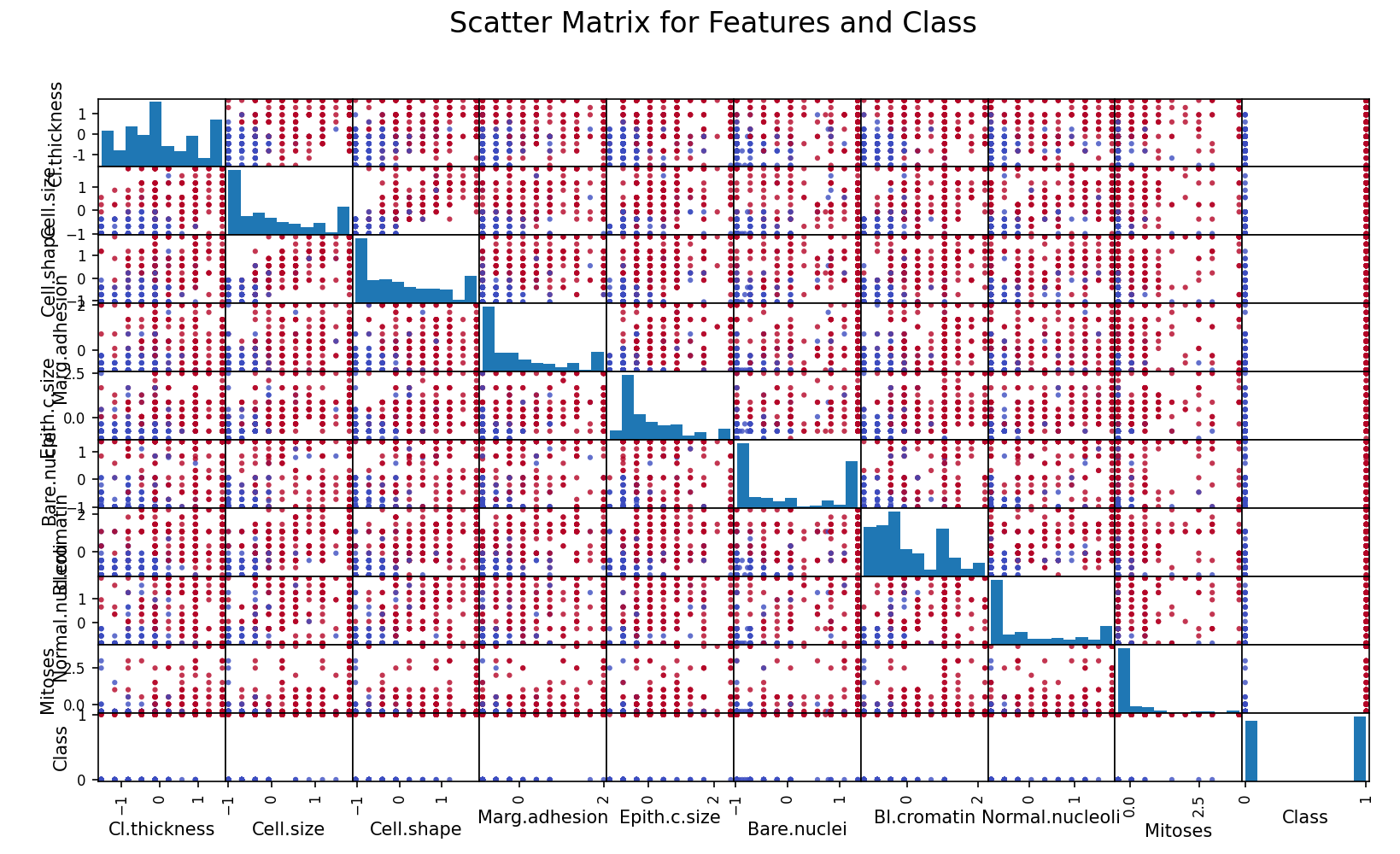
Week 10 Lab Report

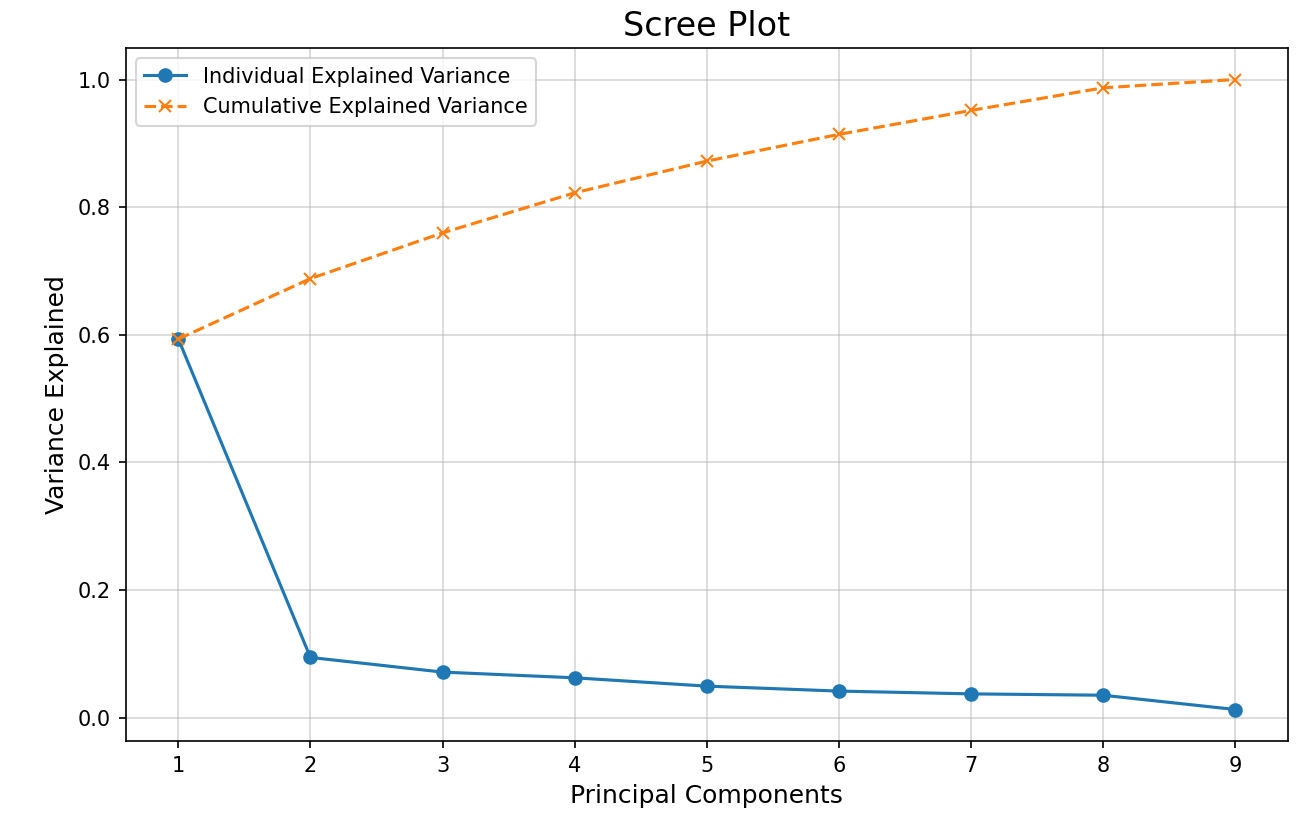
This dataset is related to breast cancer diagnosis; it contains features that represent characteristics of tumours such as cell thickness, cell size and cell shape with 460 instances. The target feature is the class column, which indicates whether the tumour is benign or malignant. The .map() function maps “benign” to 0 and “malignant” to 1. To clean up the data, I used the KNN imputation of 5 neighbours to all continuous variables if there were missing values. If these rows were to be dropped, valuable patient information would be discarded, leading to a less representative dataset. Then, standard scaling was used to ensure that all the features contribute equally to the model. The ID column was then used as the index so that it is not in the feature space, and an end-user can obtain a cell ID’s information without remembering the position.

# PCA

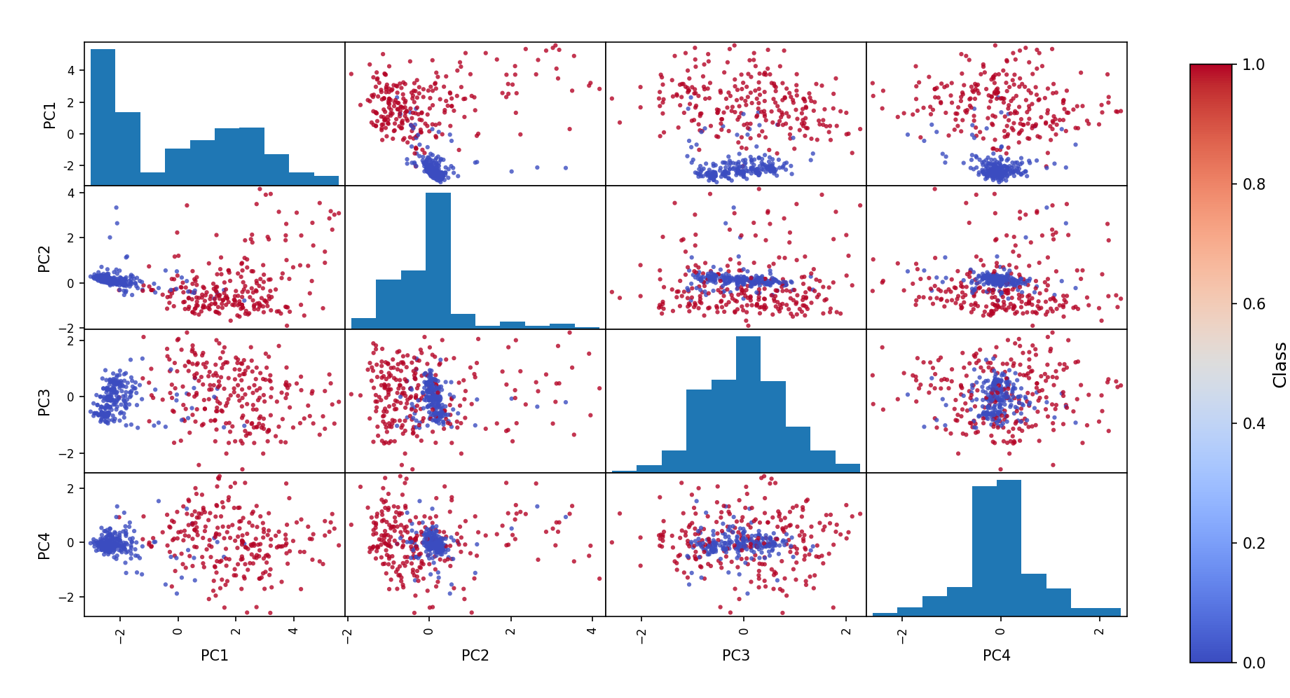
This dataset has nine features, which makes it challenging to identify trends in the data or determine whether a patient has a malignant or benign tumour. The scatter matrix below contains 100 charts.



So, to reduce dimensionality, we can conduct PCA, which takes different weightings of each feature that best explains the variance. We can use a Scree Plot to show the cumulative sum of the explained variance to determine the optimal number of principal components needed to capture most of the dataset's variance. This helps in identifying the "elbow point," where adding more components results in diminishing returns, enabling effective dimensionality reduction while preserving the most critical information. The elbow point seems to be at four components, meaning any further increase in n\_components would yield diminishing returns.

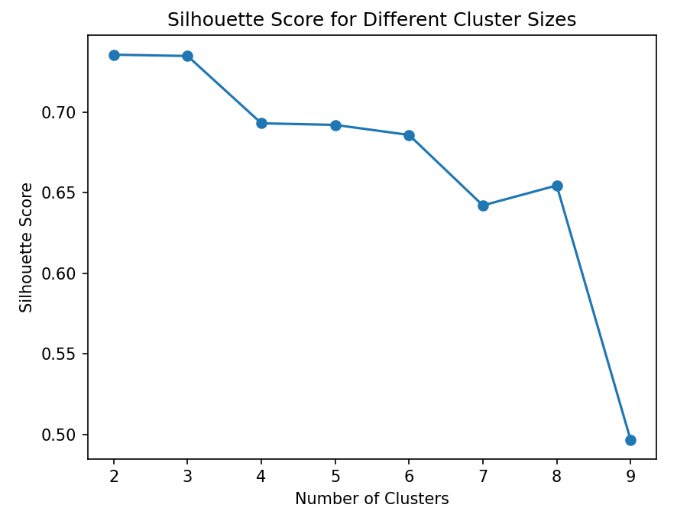
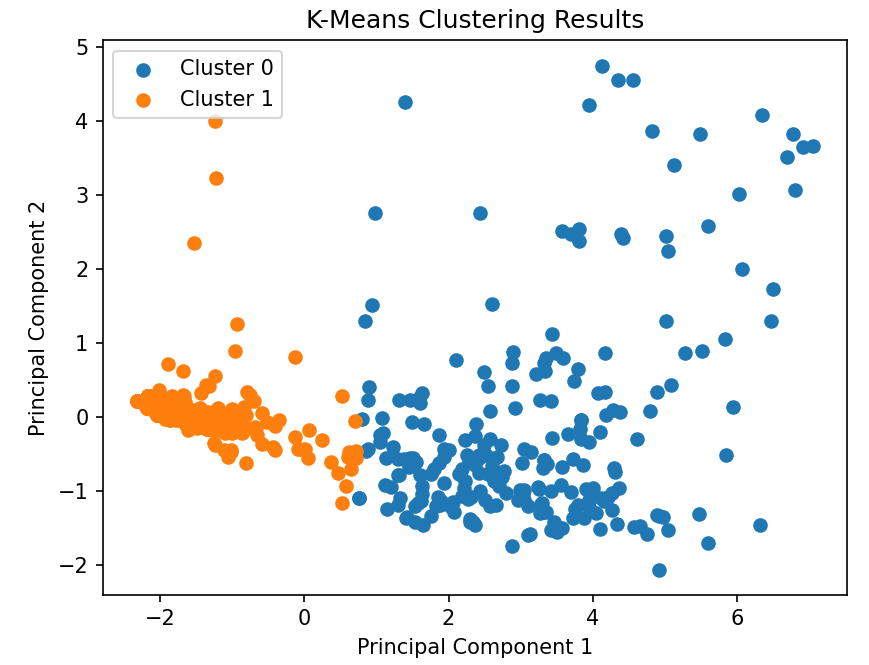


A scatter matrix of the four principal components allows us to best separate the classes, blue meaning benign and red being malignant. I chose PC1 and PC2 in the models as this produced the best separation.

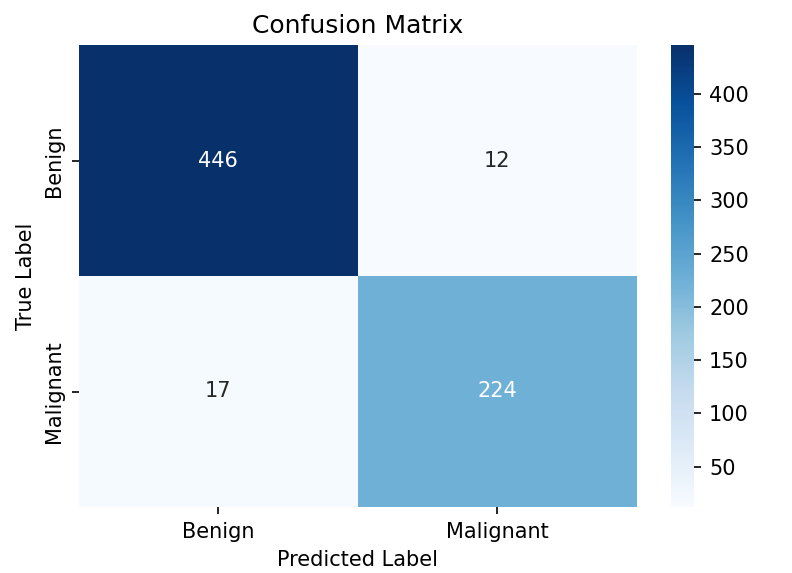


The table below shows the weights for the first four components for each of the dataset’s features. These weights reflect how much each feature contributes to the variance explained by that principal component. A highly positive weight for a particular feature compared to others in the same Principal Component indicates that this feature is a significant contributor to the direction of variance in the data, while a highly negative value contributes to the opposite direction.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Clthickness** | **Cell.size** | **Cell.shape** | **Marg.adhesion** | **Epith.c.size** | **Bare.nuclei** | **Bl.cromatin** | **Normal.nucleoli** | **Mitoses** |
| **PC1** | 0.293572 | 0.391485 | 0.386641 | 0.329308 | 0.3342 | 0.325746 | 0.353571 | 0.33329 | 0.220739 |
| **PC2** | -0.15094 | -0.02812 | -0.07083 | -0.05131 | 0.230615 | -0.31348 | -0.2414 | 0.054171 | 0.869548 |
| **PC3** | -0.84067 | 0.000546 | -0.05729 | 0.472265 | 0.132358 | 0.011889 | 0.165954 | 0.09436 | -0.11334 |
| **PC4** | -0.11787 | 0.214774 | 0.191083 | -0.4409 | 0.297189 | -0.54719 | 0.04226 | 0.467734 | -0.31746 |

If the class was not taken as the target and K-Clustering was performed after PCA transformation, the following chart shows the silhouette score for different cluster sizes and the clustering for 2 clusters which scored highest.

The following confusion matrix compares how well the K-Means clusters the classes compared to the true labels:



As we can see, the clusters for benign and malignant tumours are well-defined, which bodes well for our predictive models.

# 5 Models:

Before training the models, the data had to be prepared, including dropping ID, which is used as an identifier, and Class, which is a target. The dataset underwent a PCA transformation and was reduced to two components for visualisation. The data was then test-train split 70/30. This code used a parameter grid to define the grid search parameters for the models:

**KNN:**

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Values** | **What does it change?** |
| n\_neighbors | 3, 5, 7, 9, 11, 15 | Number of neighbors to use |
| weights | uniform, distance | Weight function used in prediction |
| metric | euclidean, manhattan, chebyshev, minkowski | for distance computation |

**Linear Logistic Regression**

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Values** | **What does it change?** |
| penalty | l1, l2, elasticnet, none | Specify the norm of the penalty |
| C | 0.01, 0.1, 1, 10, 100 | Inverse of regularization strength |
| solver | lbfgs, liblinear, saga | Algorithm to use in the optimization problem |
| Max citer | 100, 200, 500, 1000 | Maximum number of iterations taken for the solvers to converge. |
| l1 ratio | 0.1, 0.5, 0.9 |  |

**Random Forest**

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Values** | **What does it change?** |
| n\_estimators | 50, 100, 200, 500 | The number of trees in the forest. |
| max\_depth | None, 10, 20, 30, 50 | The maximum depth of the tree. |
| min\_samples\_split | 2, 5, 10 | The minimum number of samples required to split an internal node |
| min\_samples\_leaf | 1, 2, 4 | The minimum number of samples required to be at a leaf node. |
| max\_features | sqrt, log2, None | The number of features to consider when looking for the best split |
| bootstrap | True, False | Whether bootstrap samples are used when building trees. |
| criterion | gini, entropy, log\_loss | The function to measure the quality of a split |

**SVM Poly**

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Values** | **What does it change?** |
| C | 0.01, 0.1, 1, 10, 100 | Regularization parameter |
| Degree | 2, 3, 4, 5 | Polynomial degree |
| coef0 | 0, 0.5, 1, 2 | Independent term in polynomial kernel |
| gamma | scale, auto | Kernel coefficient |

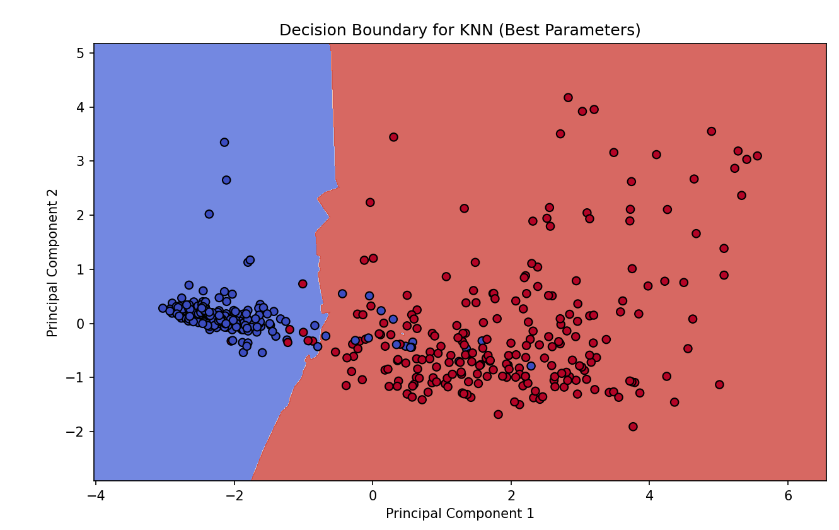
**SVM RBF**

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Values** | **What does it change?** |
| C | 0.008, 0.009, 0.01 | Regularization parameter |
| Gamma | 0.1, 0.09, 0.08, 0.07, scale, auto | Kernel coefficient |

# Best Parameters for:

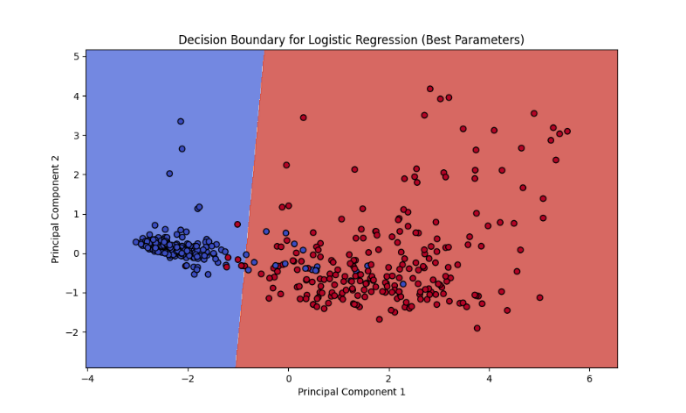
## KNN:

{'metric': 'Euclidean', 'n\_neighbors': 5, 'weights': 'uniform'}



|  |  |  |
| --- | --- | --- |
| **Label** | Benign | Malignant |
| **Precision** | 0.96774 | 0.92105 |
| **Recall** | 0.90909 | 0.97222 |
| **F1 Score** | 0.9375 | 0.94595 |
| **Support** | 66 | 72 |
| **Train Accuracy** | 0.95639 | 0.95639 |
| **Test Accuracy** | 0.94203 | 0.94203 |

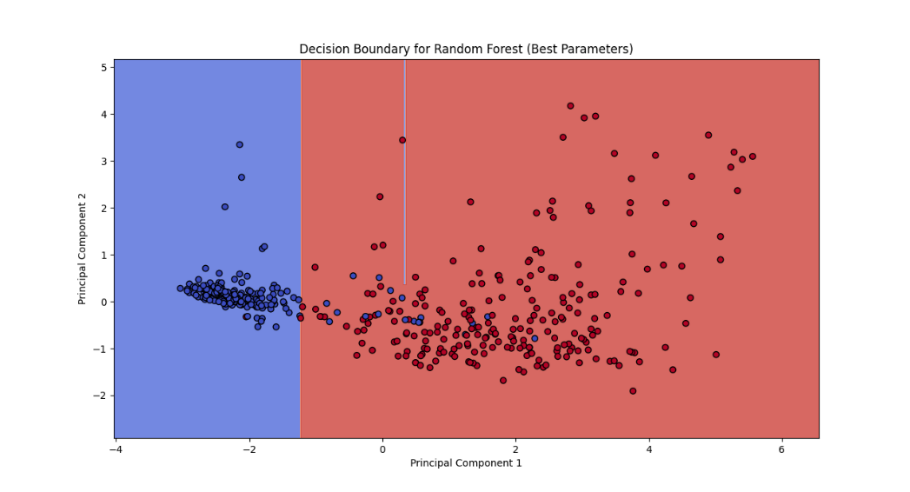
## Linear Logistic Regression:

{'C': 0.0002, 'l1\_ratio': 0.1, 'max\_iter': 100, 'penalty': 'l2', 'solver': 'lbfgs'}

|  |  |  |
| --- | --- | --- |
| **Label** | Benign | Malignant |
| **Precision** | 0.98305 | 0.89873 |
| **Recall** | 0.87879 | 0.98611 |
| **F1 Score** | 0.928 | 0.9404 |
| **Support** | 66 | 72 |
| **Train Accuracy** | 0.96885 | 0.96885 |
| **Test Accuracy** | 0.93478 | 0.93478 |

## Random Forest:

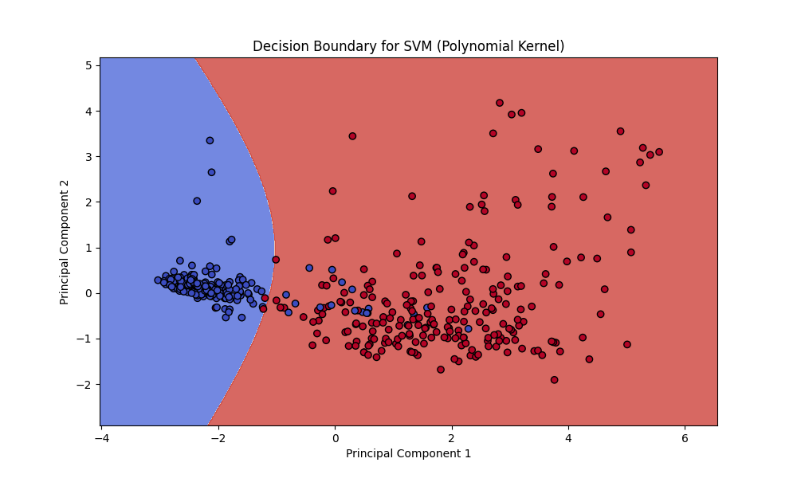
{'bootstrap': True, 'criterion': 'gini', 'max\_depth': None, 'max\_features': None, 'min\_samples\_leaf': 4, 'min\_samples\_split': 2, 'n\_estimators': 100}



|  |  |  |
| --- | --- | --- |
| **Label** | Benign | Malignant |
| **Precision** | 0.95238 | 0.92 |
| **Recall** | 0.90909 | 0.95833 |
| **F1 Score** | 0.93023 | 0.93878 |
| **Support** | 66 | 72 |
| **Train Accuracy** | 0.97508 | 0.97508 |
| **Test Accuracy** | 0.93478 | 0.93478 |

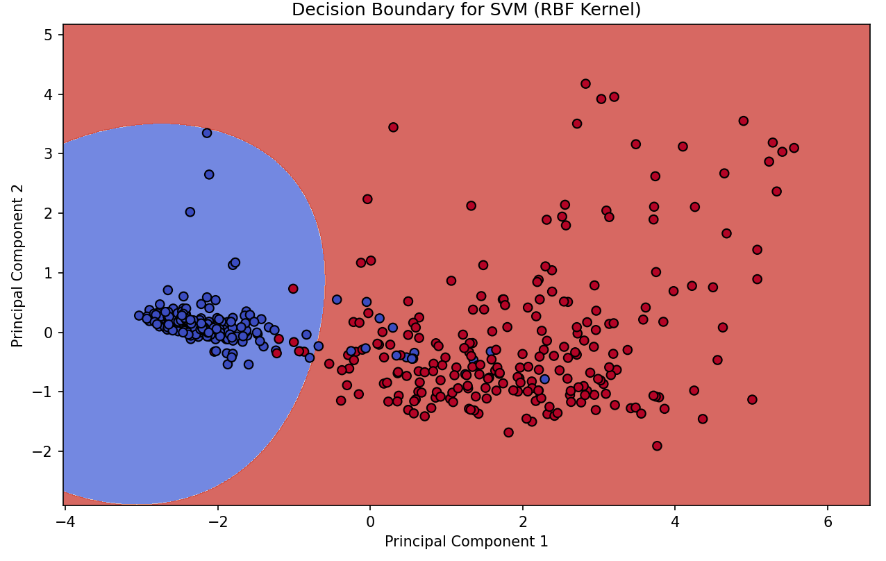
## SVM – Polynomial:

{'C': 0.008, 'coef0': 2, 'degree': 4, 'gamma': 'auto'}



|  |  |  |
| --- | --- | --- |
| **Label** | Benign | Malignant |
| **Precision** | 0.96721 | 0.90909 |
| **Recall** | 0.89394 | 0.97222 |
| **F1 Score** | 0.92913 | 0.9396 |
| **Support** | 66 | 72 |
| **Train Accuracy** | 0.96573 | 0.96573 |
| **Test Accuracy** | 0.93478 | 0.93478 |

## SVM – RBF:

{'C': 0.008, 'gamma': 0.07}

|  |  |  |
| --- | --- | --- |
| **Label** | Benign | Malignant |
| **Precision** | 0.98039 | 0.81609 |
| **Recall** | 0.75758 | 0.98611 |
| **F1 Score** | 0.8547 | 0.89308 |
| **Support** | 66 | 72 |
| **Train Accuracy** | 0.90966 | 0.90966 |
| **Test Accuracy** | 0.87681 | 0.87681 |

# Overfitting vs Underfitting

To classify whether overfitting vs underfitting, we must compare the train vs test accuracies.

If the training accuracy is low, the model is underfitting. If the training accuracy is far greater than the test, the model is overfitting and thus not generalising well.

| **Model** | **Train Accuracy** | **Test Accuracy** | **Observation** |
| --- | --- | --- | --- |
| **KNN** | 0.9564 | 0.9420 | Train and test accuracies are close, indicating no overfitting or underfitting. |
| **Logistic Regression** | 0.9688 | 0.9348 | Slightly higher train accuracy; no evidence of overfitting. |
| **Random Forest** | 0.9751 | 0.9348 | Higher train accuracy but consistent test accuracy; no significant overfitting. |
| **SVM (Polynomial Kernel)** | 0.9657 | 0.9348 | Well-aligned train and test accuracies, no overfitting. |
| **SVM (RBF Kernel)** | 0.9097 | 0.8768 | Lower test accuracy compared to train accuracy, showing slight overfitting. |

# Comparing F1 and Test Accuracies

Comparing F1 scores and test accuracies helps evaluate classification models, especially for imbalanced datasets. While test accuracy measures overall correctness, F1 balances precision (how many predicted positives are correct) and recall (how many actual positives are correctly identified), highlighting the model's performance on positive predictions. A close alignment between F1 and accuracy indicates balanced performance, while a gap may reveal issues with precision or recall that require further attention.

| **Model** | **Test Accuracy** | **F1 Score (Benign)** | **F1 Score (Malignant)** | **Macro F1 Score** |
| --- | --- | --- | --- | --- |
| **KNN** | 0.9420 | 0.9375 | 0.9459 | 0.9417 |
| **Logistic Regression** | 0.9348 | 0.9280 | 0.9404 | 0.9342 |
| **Random Forest** | 0.9348 | 0.9302 | 0.9388 | 0.9345 |
| **SVM (Polynomial Kernel)** | 0.9348 | 0.9291 | 0.9396 | 0.9344 |
| **SVM (RBF Kernel)** | 0.8768 | 0.8547 | 0.8931 | 0.8739 |

KNN is the best model, with the highest test accuracy and macro score, showing that it is great at predicting both classes. Poly SVM, Logistic Regression, and Random Forest perform similarly but slightly worse than KNN. The weakest model is Radial SVM, which underperforms in both accuracy and F1 scores, indicating that it is not well suited for this dataset.