Faculty of Engineering and Technology

Electrical and Computer Engineering Department

Machine Learning and Data Science (ENCS5341)

Assignment #3 Report

Prepared by:

Student's Name: Abdalkarim Eiss **Student's Number**: 1200015

Student's Name: Razi Atyani **Student's Number**: 1200028

Instructor: Dr. Yazan Abu Farha && Dr. Ismail Khater

Section: 1 & 2

Date: 27/12/2024

Abstract

In this assignment, we investigate and compare the performance of multiple supervised machine learning models on a classification task using the breast cancer dataset from Scikit-learn. This dataset contains features related to tumor characteristics and aims to classify tumors as malignant or benign. The process begins with data cleaning and preprocessing, followed by applying several machine learning models: K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Logistic Regression, and Ensemble Methods. Various configurations are tested for each model, such as different distance metrics for KNN, kernels for SVM, regularization techniques for Logistic Regression, and bagging/boosting strategies for Ensemble Methods. The models are evaluated using key classification metrics, including Accuracy, Precision, Recall, F1-Score, and Receiver Operating Characteristic - Area Under Curve (ROC-AUC), providing a comprehensive analysis of their strengths and weaknesses for this particular task.

Table of Contents

List of Tables

List of Figures

1 About Dataset

The breast cancer dataset is a popular dataset used for machine learning and statistical analysis tasks. It contains features of breast tumors and a target variable indicating whether a tumor is benign or malignant. The dataset contains 30 numerical feature columns that describe the characteristics of the tumor using values like the mean and standard deviation, it also contains 1 label column to indicate the tumor type, with 569 rows, reach representing a tumor sample. Before passing the dataset to the models, standardization using Standard Scaler and PCA Dimensionality reduction technique were applied, to standardize the data and lower the number of features.

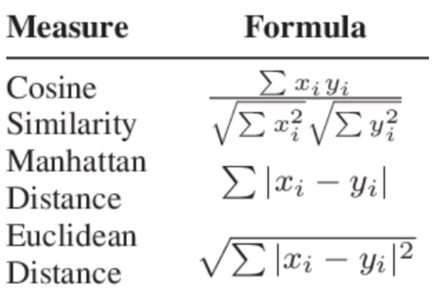
2 Models

2.1 K-Nearest Neighbor (KNN)

K-Nearest Neighbors (KNN) is a simple, supervised machine learning algorithm used for classification and regression by identifying the 'k' nearest data points based on a distance metric and making predictions based on the majority class or average value. Commonly used distance metrics in KNN include Euclidean distance, Manhattan distance, and Cosine similarity.

Euclidean distance measures the straight-line distance between two points and is best suited for continuous data with similar feature scales. **Manhattan distance** calculates the sum of absolute differences between coordinates and is ideal for categorical or discrete data, or when movement is restricted to grid-like paths. **Cosine similarity** focuses on the direction of vectors rather than their magnitude, making it useful for text data where orientation is more important than length. Each metric is chosen based on the data type and the problem at hand.

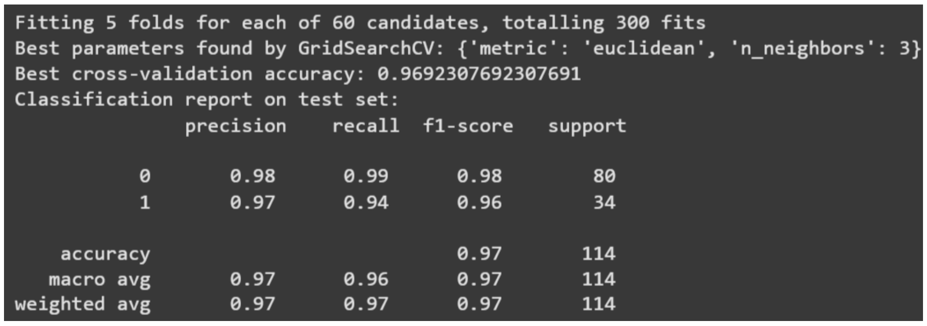
Figure 1 shows the Distances Formula.



Measure	Formula
Cosine Similarity	$\frac{\sum x_i y_i}{\sqrt{\sum x_i^2}\sqrt{\sum y_i^2}}$
Manhattan Distance	$\sum \lvert x_i - y_i \rvert$
Euclidean Distance	$\sqrt{\sum \lvert x_i - y_i \rvert^2}$

Figure 1: Distance Metrics Formulas

Figure 2 shows the KNN results.



```
Fitting 5 folds for each of 60 candidates, totalling 300 fits
Best parameters found by GridSearchCV: {'metric': 'euclidean', 'n_neighbors': 3}
Best cross-validation accuracy: 0.9692307692307691
Classification report on test set:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98        80
           1       0.97      0.94      0.96        34

    accuracy                           0.97       114
   macro avg       0.97      0.96      0.97       114
weighted avg       0.97      0.97      0.97       114
```

Figure 2: KNN Results

The approach in the code involves performing hyperparameter tuning for the K-Nearest Neighbors (KNN) algorithm using GridSearchCV. The tuning process explores different values for the number of neighbors (n_neighbors) from 1 to 20, alongside three distance metrics: Euclidean, Manhattan, and Cosine. The model is evaluated using 5-fold cross-validation, with accuracy as the scoring metric. After fitting the model with PCA-transformed training data, the best combination of hyperparameters is identified. The best KNN model is then evaluated on the test set, providing performance metrics such as accuracy, precision, recall, F1-score, and ROC AUC results. Additionally, the code individually assesses the performance of each distance metric, identifying the optimal K value for each and plotting confusion matrices to visualize the model's performance.

Figure 3 shows the performance metric for KNN with PCA data.

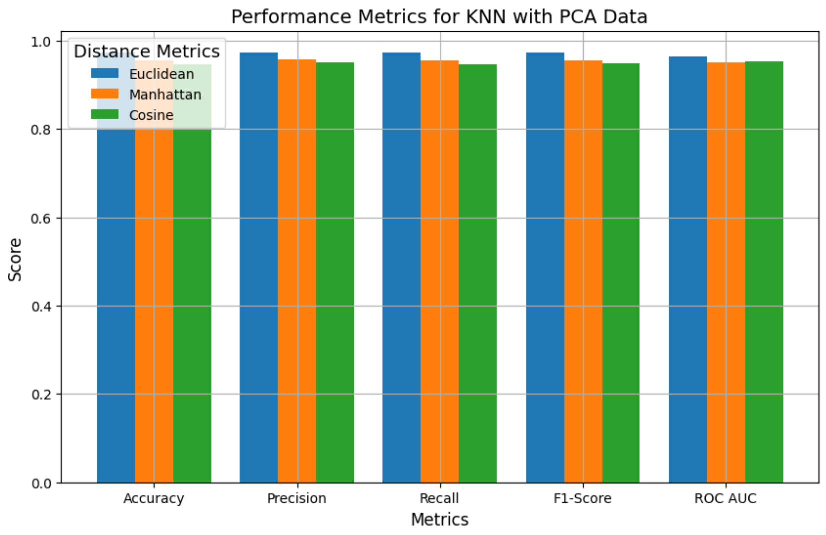


Figure 3: Performance Metric for KNN with PCA

The results show that the choice of distance metric significantly affects KNN's performance. Among the three metrics, Euclidean distance achieved the highest accuracy of 97.37%, with balanced precision, recall, and F1-score, making it the most effective for this dataset. Euclidean distance, which considers both magnitude and direction, aligns well with the structure of the PCA-transformed data. In contrast, Cosine similarity performed slightly worse with 94.74% accuracy, indicating that magnitude differences are important in this case. Manhattan distance showed intermediate performance with 95.61% accuracy. In terms of ROC AUC, Euclidean distance led with 0.9643, indicating strong class separation, followed by Cosine (0.9540) and Manhattan (0.9518). These results highlight Euclidean distance as the best performer for this dataset.

The optimal value of **K** varied depending on the distance metric. For Euclidean distance, the best K value was **3**, which allowed the model to focus on the local neighborhood, resulting in precise and robust decision boundaries without overfitting. In contrast, Cosine similarity performed best with a larger K value of **9**, indicating that a broader neighborhood was needed to achieve reliable classification, likely due to the metric's sensitivity to feature orientations. For Manhattan distance, K=5 produced the best results, balancing the need for local and global patterns in the data. These findings emphasize the importance of selecting the right combination of distance metric and K value, with Euclidean distance and K=3 emerging as the most effective pair for this dataset.

2.2 Logistic Regression

Logistic regression is a supervised algorithm commonly used for binary and multi-class classification. It predicts probabilities using the sigmoid function, which maps outputs to a range of 0 to 1. Regularization techniques like L1 (Lasso), which aids in feature selection by shrinking some coefficients to zero, and L2 (Ridge), which ensures smooth solutions by penalizing large coefficients, are often applied to enhance generalization and prevent overfitting.

Effective for linearly separable classes, logistic regression is interpretable, as its coefficients indicate the strength and direction of predictor-target relationships. While it assumes linearity between predictors and log-odds, limiting its ability to capture non-linear patterns, it remains a robust, scalable, and widely-used baseline model, especially when combined with regularization.

Figure 4 shows the Sigmoid Function.

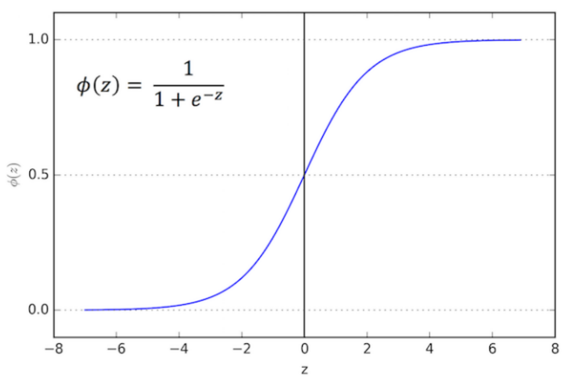
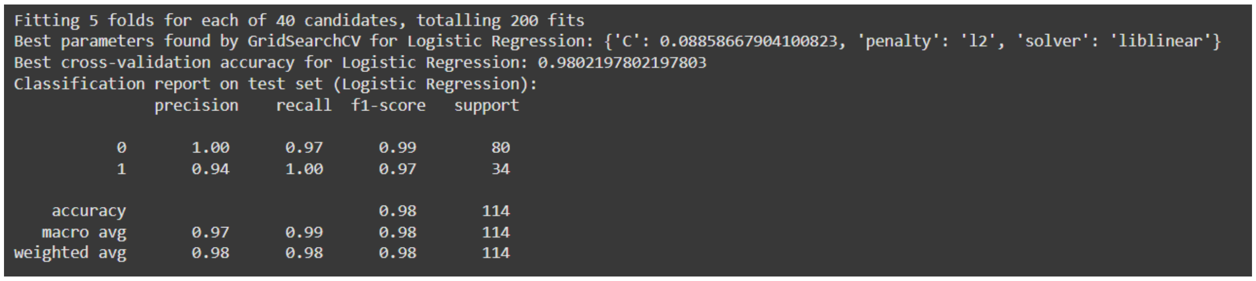


Figure 4: Sigmoid Function

Figure 5 shows the Logistic regression results.



```
Fitting 5 folds for each of 40 candidates, totalling 200 fits
Best parameters found by GridSearchCV for Logistic Regression: {'C': 0.08858667904100823, 'penalty': 'l2', 'solver': 'liblinear'}
Best cross-validation accuracy for Logistic Regression: 0.9802197802197803
Classification report on test set (Logistic Regression):
              precision    recall  f1-score   support

           0       1.00      0.97      0.99        80
           1       0.94      1.00      0.97        34

    accuracy                           0.98       114
   macro avg       0.97      0.99      0.98       114
weighted avg       0.98      0.98      0.98       114
```

Figure 5: Logistic Regression Results

The code implements a comprehensive evaluation of logistic regression using L1 (Lasso) and L2 (Ridge) regularization through GridSearchCV to optimize the regularization strength (C) and identify the best-performing model based on cross-validation accuracy. PCA-transformed training data is used to train the logistic regression model, and the best hyperparameters are determined. Performance is evaluated on the test set using metrics such as accuracy, precision, recall, F1-score, ROC AUC results and a confusion matrix for each regularization type. Additionally, the performance of logistic regression is compared with KNN in terms of key metrics, and the results are visualized using bar plots for a clear comparison of the two models. The approach ensures robust model selection and comparative analysis.

Figure 6 shows the Comparison between KNN and Logistic Regression results.

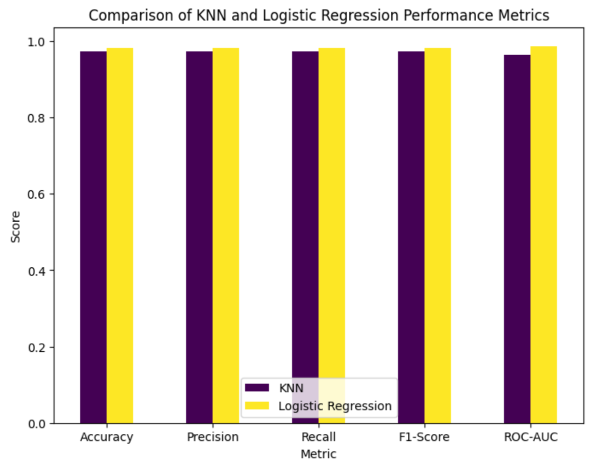


Figure 6: Comparison between KNN and Logistic Regression results

The results show that **Logistic Regression with L2 regularization** (Ridge) outperforms **KNN** in all key performance metrics, including **accuracy, precision, recall, F1-score**, and **ROC-AUC**. With a test accuracy of **98.25%**, logistic regression demonstrates its ability to correctly classify samples while maintaining a good balance between precision and recall, as evidenced by an F1-score of **98.26%** and a **ROC-AUC of 0.9875**. The L2 regularization with an optimal **C value of 0.0886** helps the model generalize well without overfitting, making it a reliable choice for this dataset. In contrast, **KNN** performs well with an accuracy of **97.37%** and a **ROC-AUC of 0.9643**, but its proximity-based classification is more sensitive to noise and high-dimensional spaces, resulting in marginally lower performance compared to logistic regression.

Although **Logistic Regression with L1 regularization** (Lasso) showed slightly lower results with **94.74% accuracy**, it still delivered strong metrics, including **95.53% precision** and **94.85% F1-score**, alongside a **ROC-AUC of 0.9625**. However, logistic regression with L2 regularization provided superior performance across all metrics, highlighting its strength in handling linear decision boundaries and regularization. Overall, **Logistic Regression with L2 regularization** emerges as the preferred model due to its robustness, higher generalization, and simplicity compared to KNN, making it the optimal choice for this classification task.

2.3 Support Vector Machine (SVM)

Support Vector Machines (SVM) use kernels to transform data into higher-dimensional spaces, enabling solutions to non-linear problems. The linear kernel, suited for linearly separable data, is computationally efficient and interpretable, as feature coefficients indicate their influence. It works well for datasets with simple, linear decision boundaries and serves as a baseline for classification tasks.

For more complex data, the polynomial kernel introduces non-linearity by modeling feature interactions of a specified degree, while the RBF kernel maps data into infinite-dimensional space using a Gaussian function, effectively handling clusters and non-linear patterns. The RBF kernel requires careful tuning of the gamma parameter to avoid overfitting. Both kernels are flexible and capable of handling complex decision boundaries, making them popular for diverse SVM applications.

Figure 7 shows the Comparison between linear and Polynomial.

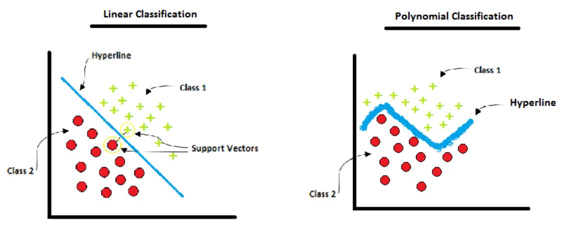
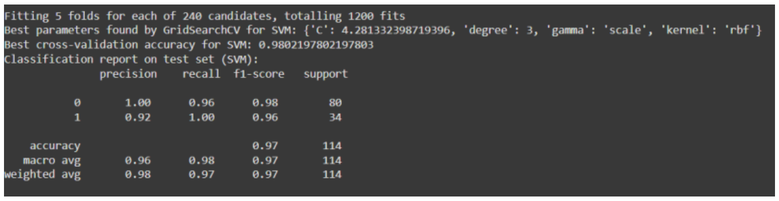


Figure 7: Comparison between linear and Polynomial

Figure 8 shows the Results of SVM.



```
Fitting 5 folds for each of 240 candidates, totalling 1200 fits
Best parameters found by GridSearchCV for SVM: {'C': 4.281332398719396, 'degree': 3, 'gamma': 'scale', 'kernel': 'rbf'}
Best cross-validation accuracy for SVM: 0.9802197802197803
Classification report on test set (SVM):
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        80
           1       0.92      1.00      0.96        34

    accuracy                           0.97       114
   macro avg       0.96      0.98      0.97       114
weighted avg       0.98      0.97      0.97       114
```

Figure 8: SVM Results

The approach involves performing a comprehensive comparison of different SVM kernels (linear, polynomial, and RBF) using GridSearchCV to find the optimal hyperparameters (C and gamma) through cross-validation. The model is trained with PCA-transformed data, and performance is evaluated on a test set using multiple metrics, including accuracy, precision, recall, F1-score, and AUC (from the ROC curve). For each kernel, the best hyperparameters are determined, and the corresponding model's performance is assessed, including plotting ROC curves and confusion matrices. The results are collected and displayed in a bar plot for an easy comparison of the kernels' performance across all metrics.

Figure 9 shows the Results of SVM.

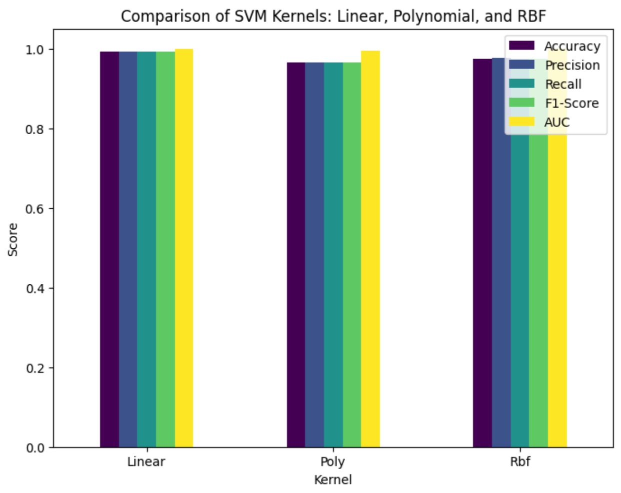


Figure 9: Comparison of kernels

The choice of kernel has a significant impact on the performance of the Support Vector Machine (SVM) model. The linear kernel provided the best overall results, achieving the highest accuracy (99.12%), precision, recall, and F1-score, all around 99.1%, along with an AUC of 0.9996. This suggests that the dataset likely has a linear or near-linear decision boundary, making the linear kernel the most effective for this problem. The RBF kernel, with an accuracy of 97.37% and similar performance metrics, performed well as it can capture more complex, non-linear patterns, though it slightly lagged behind the linear kernel in terms of accuracy. The polynomial kernel, with an accuracy of 96.49%, showed a good performance but was outpaced by both the linear and RBF kernels. It still performed well in terms of AUC (0.9949), indicating it can handle non-linear decision boundaries, but it was not as effective as the other two kernels. Overall, the linear kernel proved to be the best choice for this particular dataset, while the RBF kernel offered flexibility for non-linear relationships, and the polynomial kernel was more suitable for highly complex, non-linear problems. In summary, the best-performing model was the **linear kernel** with **C = 0.0886** and **gamma = 'scale'**, followed by the **RBF kernel** with **C = 4.2813** and **gamma = 'scale'**, and lastly the **polynomial kernel** with **C = 0.6158** and **gamma = 'auto'.**

2.4 Ensemble Methods

Figure 10 shows the comparison between AdaBoost and Random Forest.

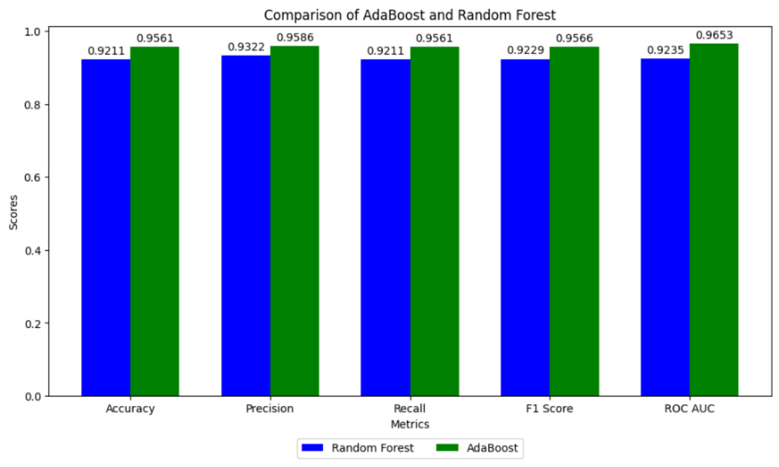


Figure 10: The Comparison Between AdaBoost and Random Forest

2.4.1 Boosting: AdaBoost

Boosting is an advanced ensemble learning technique that aims to improve the performance of weak learners by combining them iteratively to form a strong predictive model. A weak learner, such as a simple decision tree, performs slightly better than random guessing. Boosting works by training these weak learners sequentially, with each one attempting to correct the errors of its predecessor. By assigning higher weights to misclassified samples, subsequent models focus more on these challenging cases, resulting in improved predictions. This process effectively reduces bias and variance, striking a balance that enhances the overall model performance. Boosting is particularly powerful for handling complex datasets where a single model may struggle to achieve high accuracy.

In this approach, the AdaBoost algorithm, a popular boosting method, was implemented. Adaptive weights were assigned to data points, with harder-to-predict samples being emphasized as the algorithm progressed. Key hyperparameters, such as the number of estimators and learning rate, were optimized using a grid search to maximize performance. An accuracy of 95.61%, precision of 95.86%, recall of 95.61%, F1-score of 95.66%, and an ROC AUC of 96.03% were achieved by the optimized AdaBoost model on the test dataset. These metrics demonstrate the strength of boosting in creating a robust and efficient model, highlighting its ability to adapt and excel in challenging classification tasks.

2.4.2 Bagging: Random Forest

Bagging or Bootstrap Aggregating is an ensemble learning method that builds multiple models (base learners) from random subsets of the data and then aggregates their predictions to reduce variance and prevent overfitting. In this approach, the base model used is a Random Forest, which itself is an ensemble of decision trees. Bagging with Random Forest enhances the performance by creating multiple Random Forest classifiers trained on different bootstrapped subsets of the data. The predictions of these classifiers are combined through majority voting or averaging to give a final prediction.

In this implementation, the hyperparameter tuning for the Bagging Random Forest model is performed using GridSearchCV with cross-validation. The tuning involved optimizing parameters such as the number of base estimators, the number of trees in each Random Forest, and parameters like maximum depth, minimum samples for splitting, and whether bootstrap sampling was used. The tuned Bagging Random Forest model achieved competitive results with an accuracy of 92.45%, precision of 92.67%, recall of 92.45%, F1-score of 92.56%, and an ROC AUC of 92%. These results demonstrate the effectiveness of the Bagging Random Forest approach in improving the robustness of the model while maintaining high classification performance.

2.4.3 Ensemble Methods and Individual Models Comparison

Among the ensemble methods, boosting (AdaBoost) performed better than Bagging (Random Forest), achieving higher accuracy, precision, recall, F1-score and ROC AUC. Boosting works by iteratively focusing on correcting errors made by previous models, which allows it to enhance performance on difficult-to-predict samples. In contrast, Bagging, like Random Forest, reduces variance by averaging predictions across multiple models but does not directly correct errors. Compared to individual models like KNN, Logistic Regression and SVM, ensemble methods generally perform better, as they combine multiple weak learners to handle complex and noisy data more effectively, offering improved generalization, reduced bias, and variance. Individual models may struggle with high-dimensional, noisy datasets or non-linear relationships, whereas ensemble methods reduce overfitting and improve robustness by leveraging multiple models.

2.5 Comparison Between Best Models

Table 1 shows the comparison between best models.

Model	**Configuration**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**ROC AUC**
KNN	Best Euclidean (k=3)	0.9737	0.9736	0.9737	0.9736	0.9643
	Best Manhattan (k=5)	0.9561	0.9567	0.9561	0.9563	0.9518
	Best Cosine (k=9)	0.9474	0.9515	0.9474	0.9482	0.9540
Logistic Regression	L1 Regularization	0.9474	0.9553	0.9474	0.9485	0.9625
	L2 Regularization	0.9825	0.9834	0.9825	0.9830	0.9875
SVM	Linear Kernel	0.9912	0.9915	0.9912	0.9913	0.9996
	Poly Kernel	0.9649	0.9650	0.9649	0.9646	0.9949
	RBF Kernel	0.9737	0.9758	0.9737	0.9747	0.9996
AdaBoost	Best Parameters	0.9561	0.9586	0.9561	0.9566	0.9603
Random Forest	Best Parameters	0.9211	0.9322	0.9211	0.9229	0.9200

Table 1: Overall Results

As noted, the L2 regularization model and linear SVM model are the best.

3 Conclusion

In conclusion, the performance of various supervised machine learning models on the breast cancer dataset was evaluated and compared, with a focus on classification metrics such as Accuracy, Precision, Recall, F1-Score, and ROC-AUC. Among the models tested, the best performance was demonstrated by the Support Vector Machine (SVM) with a Linear Kernel, with the highest Accuracy (0.9912), Precision (0.9915), Recall (0.9912), F1-Score (0.9913), and ROC-AUC (0.9996), making it the most suitable model for this task. Competitive results were also delivered by Logistic Regression with L2 regularization, achieving an Accuracy of 0.9825 and ROC-AUC of 0.9875, which provided a balance of performance and interpretability. Strong performance was observed from the K-Nearest Neighbors (KNN) model, particularly with the Euclidean distance metric (k=3), with an Accuracy of 0.9737 and ROC-AUC of 0.9643, although variation in results was noted with different configurations. Satisfactory but comparatively lower performance was shown by ensemble methods such as AdaBoost and Random Forest, with AdaBoost achieving an Accuracy of 0.9561 and Random Forest scoring an Accuracy of 0.9211. Overall, the SVM with a Linear Kernel was identified as the most effective model, while KNN and Logistic Regression were recognized as strong alternatives, depending on the specific requirements of the application.