جامعة بيرزيت

BIRZEIT UNIVERSITY

Faculty of Engineering & Technology

Department of Electrical & Computer Engineering

ENCS5341 – Machine Learning and Data Since

First Semester 2024/2025

Assignment #3

Prepared by

Jalila Muadi – 1201611

Section No. 1

Instructor: Dr. Yazan Abu Farha

BIRZEIT

December 2024

Abstract

Machine learning plays a pivotal role in solving classification problems across diverse domains, including healthcare and technology. This report evaluates the performance of various machine learning algorithms on the Medical Cost Personal Datasets, aiming to predict the geographical region of residence based on individual features. Models such as K-Nearest Neighbors (KNN), Logistic Regression, Support Vector Machines (SVM) with different kernels, and ensemble methods (AdaBoost and Random Forest) were implemented and compared. A robust experimental methodology was adopted, including data preprocessing, hyperparameter tuning, and performance evaluation using metrics like accuracy, precision, recall, F1-score, and ROC-AUC. The findings highlight the strengths and weaknesses of individual models and ensemble methods, providing insights into their suitability for classification tasks.

Table of Contents

List of Figures

List of Tables

Introduction

Machine learning transforms data into insights through classification algorithms, addressing challenges in healthcare, finance, and technology. This assignment compares core classification methods to evaluate their performance and applicability.

1. K-Nearest Neighbors (KNN)

Is a non-parametric, supervised learning algorithm widely used for both classification and regression tasks. It classifies data points based on the proximity of their neighbors, following the principle that similar data points are often located near one another. For classification, KNN assigns a class label to a new data point based on a majority vote of its k-nearest neighbors, while for regression, it predicts a value based on the average of these neighbors. Unlike many algorithms, KNN does not involve a training phase; instead, it relies on the entire dataset during prediction, categorizing it as a lazy learning or instance-based method. Despite its simplicity and effectiveness for smaller datasets, KNN can become computationally expensive as the dataset grows, making it better suited for smaller-scale problems like recommendation systems, intrusion detection, and pattern recognition. [1]

2. Logistic Regression

Logistic Regression is a statistical and machine learning method used for binary classification problems, predicting the probability of an event's occurrence. Unlike linear regression, which predicts continuous outcomes, logistic regression predicts categorical outcomes by modeling the relationship between independent variables and a dependent variable using the logistic function. This function transforms probabilities into a logit scale, ensuring the output values range between 0 and 1. The coefficients in logistic regression are estimated using Maximum Likelihood Estimation (MLE), and regularization techniques such as L1 and L2 are often applied to prevent overfitting. Widely used in domains like fraud detection, disease prediction, and customer churn analysis, logistic regression remains a simple yet powerful tool for understanding and predicting categorical outcomes. [2]

3. Support Vector Machines (SVM) with kernels

Support Vector Machines (SVMs) are versatile supervised learning models used for classification and regression tasks. A key feature of SVMs is their ability to utilize different kernel functions to handle various data distributions. Kernels enable SVMs to find optimal decision boundaries, even in cases where data is not linearly separable in the original feature space. Commonly used SVM kernels include: [3]

- **Linear Kernel**: Suitable for linearly separable data, it computes the dot product between input vectors without transforming the feature space.
- **Polynomial Kernel**: Capable of modeling interactions between features, it computes the similarity of input vectors in terms of polynomial functions, allowing for more complex decision boundaries.
- **Radial Basis Function (RBF) Kernel**: Also known as the Gaussian kernel, it can handle complex, non-linear relationships by mapping input vectors into an infinite-dimensional space, making it effective in high-dimensional settings. [3]

4. Ensemble Methods: Boosting and Bagging

Ensemble methods enhance predictive performance by combining multiple models to produce a more robust and accurate composite model. Two widely used ensemble techniques are **Bagging** and **Boosting**. [4]

Bagging (Bootstrap Aggregating) involves training multiple instances of a base model on different subsets of the training data, created through bootstrapping (sampling with replacement). Each model is trained independently, and their predictions are aggregated—commonly by averaging for regression tasks or majority voting for classification—to form the final output. Bagging primarily aims to reduce variance and overfitting, leading to improved model stability and accuracy. A notable implementation of bagging is the Random Forest algorithm, which combines multiple decision trees to enhance performance. [4]

Boosting, in contrast, trains models sequentially, with each new model focusing on correcting the errors of its predecessors. Initially, all data points carry equal weight; subsequent models adjust weights to emphasize misclassified instances, guiding the learning process toward difficult cases. This iterative approach converts weak learners into a robust ensemble capable of achieving high accuracy. However, boosting can be more susceptible to overfitting, especially when dealing with noisy datasets. [4]

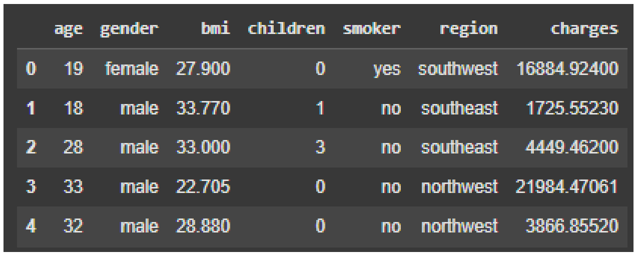
Dataset chosen

For this assignment, the **Medical Cost Personal Datasets** dataset from the ENCS5141 AI Lab course was used. The dataset contains information about individuals and their medical insurance charges. It includes seven features: [5]

- **Age**: Age of the individual.
- **Gender**: Gender of the individual (male or female).
- **BMI**: Body Mass Index, a measure of body fat based on height and weight.
- **Children**: Number of children/dependents covered by the insurance.
- **Smoker**: Whether the individual is a smoker (yes or no).
- **Region**: Geographical region of residence (northeast, northwest, southeast, southwest).
- **Charges**: Medical insurance charges billed to the individual.

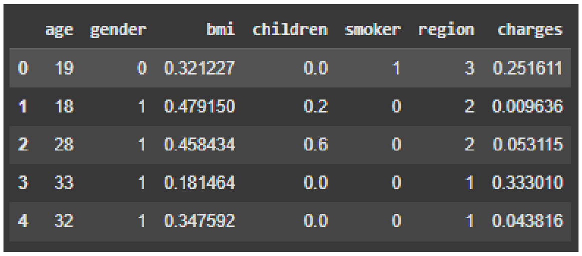
The dataset was loaded using the pandas library in Python, and an initial exploration revealed that it contains 1,338 rows and 7 columns. The dataset was found to be clean and did not require significant preprocessing, as no missing or invalid values were present. To prepare the dataset for machine learning tasks, the following steps were taken:

1. **Label Encoding**: Categorical variables such as gender, smoker, and region were converted into numerical values using label encoding to ensure compatibility with machine learning algorithms.
2. **Normalization**: Continuous variables like BMI and charges were standardized as needed to ensure balanced scaling across features.



	age	gender	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

Figure 1: Dataset Before Preparation and Preprocessing



	age	gender	bmi	children	smoker	region	charges
0	19	0	0.321227	0.0	1	3	0.251611
1	18	1	0.479150	0.2	0	2	0.009636
2	28	1	0.458434	0.6	0	2	0.053115
3	33	1	0.181464	0.0	0	1	0.333010
4	32	1	0.347592	0.0	0	1	0.043816

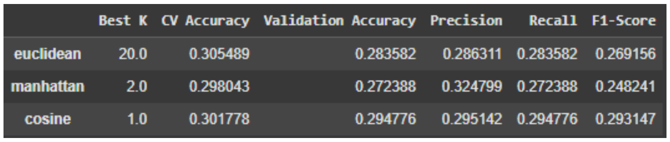
Figure 2: Dataset After Preparation and Preprocessing

Split the Data

To evaluate the performance of machine learning models, the dataset was divided into training, validation, and test sets in a 60:20:20 ratio. The training set was used for model learning, the validation set for hyperparameter tuning and optimization, and the test set for assessing the model's performance on unseen data. This division ensures that each dataset serves a distinct purpose in the modeling process, minimizing the risk of overfitting and providing a reliable evaluation metric.

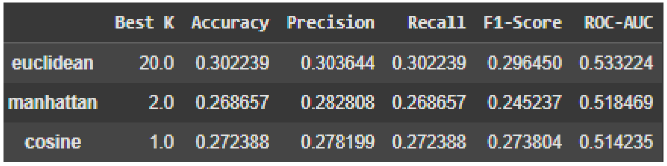
Part 1: K-Nearest Neighbors (KNN)

To identify the optimal number of neighbors (K) and evaluate the impact of distance metrics, three metrics were tested: Euclidean, Manhattan, and Cosine. Grid search combined with 5-fold cross-validation was employed to determine the best value of K for each distance metric, ensuring a robust selection process. The results of cross-validation accuracy and validation set performance provided insights into the effectiveness of each metric. Subsequently, the optimal K and metric combinations obtained from the validation phase were applied to the test set for final evaluation, allowing for a comprehensive assessment of the model's performance on unseen data.



	Best K	CV Accuracy	Validation Accuracy	Precision	Recall	F1-Score
euclidean	20.0	0.305489	0.283582	0.286311	0.283582	0.269156
manhattan	2.0	0.298043	0.272388	0.324799	0.272388	0.248241
cosine	1.0	0.301778	0.294776	0.295142	0.294776	0.293147

Figure 3: The results of cross-validation accuracy and validation set performance



	Best K	Accuracy	Precision	Recall	F1-Score	ROC-AUC
euclidean	20.0	0.302239	0.303644	0.302239	0.296450	0.533224
manhattan	2.0	0.268657	0.282808	0.268657	0.245237	0.518469
cosine	1.0	0.272388	0.278199	0.272388	0.273804	0.514235

Figure 4: Final Evaluation on Test Set

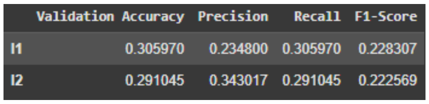
Euclidean distance consistently outperformed Manhattan and Cosine metrics across all evaluation metrics, achieving the highest test accuracy (0.302239), F1-Score (0.29645), and ROC-AUC (0.533224). This indicates that the feature relationships in the dataset align well with Euclidean geometry, making it the most effective metric for this task. In comparison, Manhattan distance demonstrated moderate performance, while Cosine distance exhibited the lowest overall performance, suggesting that these metrics were less suited for capturing the underlying patterns in the dataset. The best value of K varied by distance metric:

- For Euclidean, a higher value of K (20) worked better because it reduces overfitting by averaging over more neighbors, which might suit the feature distribution.
- For Manhattan, a smaller K (2) was optimal, likely because the dataset's class boundaries benefit from closer, local observations.
- For Cosine, K=1 yielded the best results, possibly because the cosine similarity works best with the nearest individual neighbor, rather than averaging multiple neighbors.

In result, the combination of Euclidean distance with K=20 provided the highest classification performance and is recommended as the optimal setup for this data set.

Part 2: Logistic Regression

Logistic Regression was evaluated using two regularization techniques, L1 (Lasso) and L2 (Ridge), to understand their impact on the model's performance. Regularization helps prevent overfitting by penalizing large coefficients, with L1 encouraging sparsity (some coefficients set to zero) and L2 shrinking coefficients without removing them entirely. On the validation set, the performance of Logistic Regression was measured. The results are summarized as follows:



	Validation Accuracy	Precision	Recall	F1-Score
l1	0.305970	0.234800	0.305970	0.228307
l2	0.291045	0.343017	0.291045	0.222569

Figure 5: Validation Set Performance (Logistic Regression)

Logistic Regression and K-Nearest Neighbors (KNN) were compared on the test set across several performance metrics, including Accuracy, Precision, Recall, F1-Score, and ROC-AUC.

Table 1: Comparison of Test Set Performance: Logistic Regression (L1 & L2) vs KNN with Different Distance Metrics (in test samples)

Model	Test Accuracy	Precision	Recall	F1-Score	ROC-AUC
Logistic (L1)	0.350746	0.261329	0.350746	0.273937	0.600088
Logistic (L2)	0.354478	0.374384	0.354478	0.285663	0.602323
KNN (Euclidean)	0.302239	0.303644	0.302239	0.296450	0.533224
KNN (Manhattan)	0.268657	0.282808	0.268657	0.245237	0.518469
KNN (Cosine)	0.272388	0.278199	0.272388	0.273804	0.514235

Logistic Regression with L2 regularization demonstrated the best overall performance, achieving the highest Test Accuracy (0.354478), Precision (0.374384), F1-Score (0.285663), and ROC-AUC (0.602323). This highlights its ability to generalize well and maintain a balance between precision and recall. Logistic Regression with L1 regularization also performed consistently, showing comparable Recall (0.350746) and reasonable Precision (0.261329), although slightly behind L2 in overall metrics.

Among the KNN models, the Euclidean distance variant outperformed the Manhattan and Cosine distance metrics, achieving the highest Test Accuracy and F1-Score. However, KNN models lagged behind Logistic Regression across all metrics, with significantly lower Precision, Recall, and ROC-AUC values. The ROC-AUC for KNN models peaked at 0.533224 for Euclidean distance, far below the 0.602323 achieved by Logistic Regression (L2).

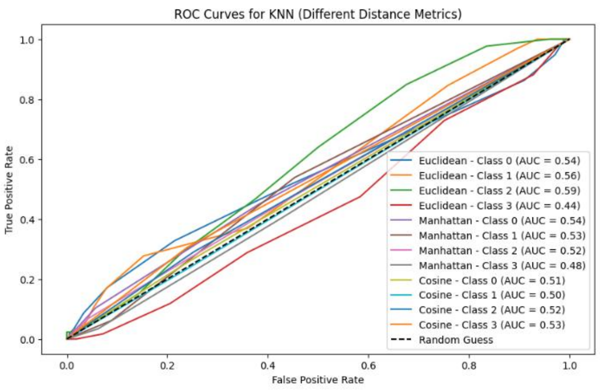


Figure 6: ROC Curves for KNN (Different Distance Metrics)

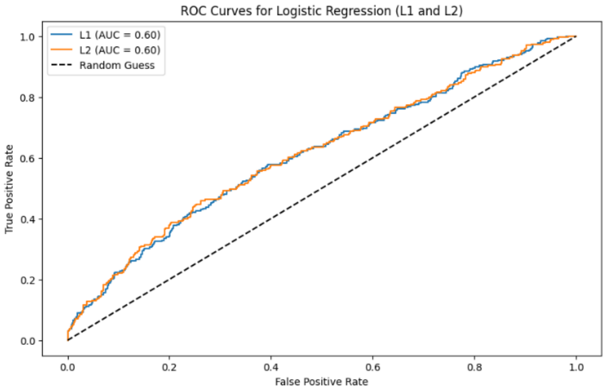
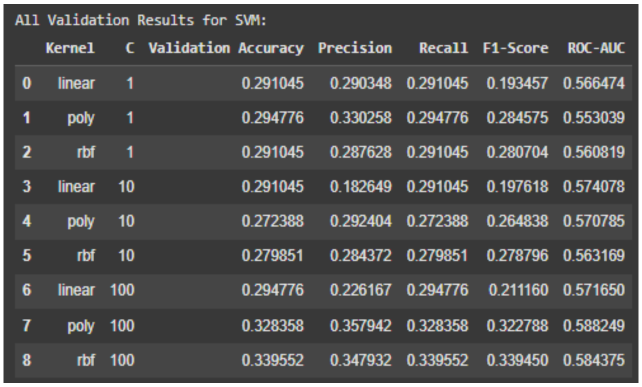


Figure 7: ROC Curves for Logistic Regression (L1 and L2)

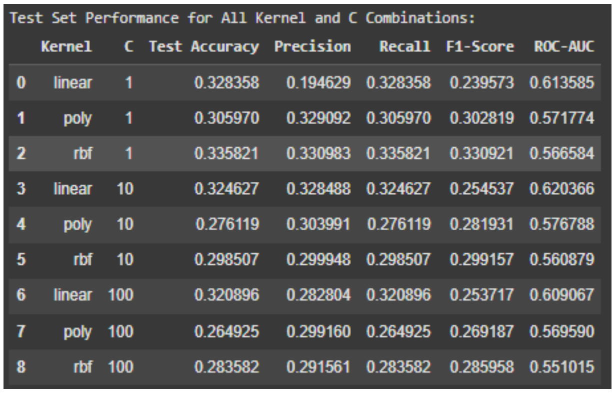
Part 3: Support Vector Machines (SVM)

The performance of Support Vector Machines (SVM) was evaluated on the test set using three kernels (**linear**, **polynomial (poly)**, and **radial basis function (rbf)**) at various values of the regularization parameter **C**.


All Validation Results for SVM:

	Kernel	C	Validation Accuracy	Precision	Recall	F1-Score	ROC-AUC
0	linear	1	0.291045	0.290348	0.291045	0.193457	0.566474
1	poly	1	0.294776	0.330258	0.294776	0.284575	0.553039
2	rbf	1	0.291045	0.287628	0.291045	0.280704	0.560819
3	linear	10	0.291045	0.182649	0.291045	0.197618	0.574078
4	poly	10	0.272388	0.292404	0.272388	0.264838	0.570785
5	rbf	10	0.279851	0.284372	0.279851	0.278796	0.563169
6	linear	100	0.294776	0.226167	0.294776	0.211160	0.571650
7	poly	100	0.328358	0.357942	0.328358	0.322788	0.588249
8	rbf	100	0.339552	0.347932	0.339552	0.339450	0.584375

Figure 8: Validation Set Performance for All Kernel and C Combinations


Test Set Performance for All Kernel and C Combinations:

	Kernel	C	Test Accuracy	Precision	Recall	F1-Score	ROC-AUC
0	linear	1	0.328358	0.194629	0.328358	0.239573	0.613585
1	poly	1	0.305970	0.329092	0.305970	0.302819	0.571774
2	rbf	1	0.335821	0.330983	0.335821	0.330921	0.566584
3	linear	10	0.324627	0.328488	0.324627	0.254537	0.620366
4	poly	10	0.276119	0.303991	0.276119	0.281931	0.576788
5	rbf	10	0.298507	0.299948	0.298507	0.299157	0.560879
6	linear	100	0.320896	0.282804	0.320896	0.253717	0.609067
7	poly	100	0.264925	0.299160	0.264925	0.269187	0.569590
8	rbf	100	0.283582	0.291561	0.283582	0.285958	0.551015

Figure 9: Test Set Performance for All Kernel and C Combinations

The linear kernel performed reasonably well at lower C values, achieving a Test Accuracy of 0.328358 and ROC-AUC of 0.613585 at C=1. This suggests it is somewhat effective at distinguishing between classes but struggles with more complex patterns. As C increased, the linear kernel showed improved performance in terms of ROC-AUC (0.620366 at C=10) but did not achieve significant improvements in accuracy or F1-Score, which remained relatively low.

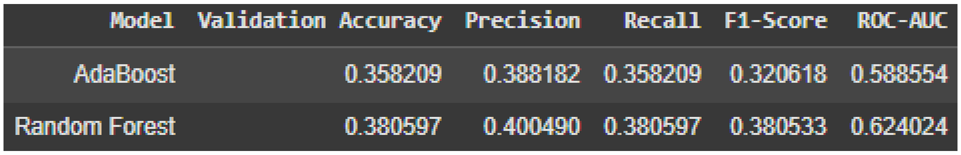
The polynomial kernel demonstrated competitive performance, particularly at C=10, where it achieved a strong ROC-AUC of 0.576788 and reasonable balance across other metrics like Precision (0.303991) and Recall (0.276119). At C=1, it achieved its highest Test Accuracy (0.305970), suggesting it captures non-linear patterns effectively. However, its performance dropped at higher C values (C=100).

The RBF kernel showed strong performance at C=1, achieving a Test Accuracy of 0.335821 and F1-Score of 0.330921. This indicates its suitability for non-linear data distributions. Like the polynomial kernel, its performance decreased at higher C values (C=100), as evidenced by a drop in ROC-AUC (0.551015) and other metrics, likely due to overfitting. The choice of kernel significantly impacts how well the SVM model fits the data:

- **Linear Kernel**: Performs adequately for datasets with simpler relationships but struggles to capture complex patterns, as evidenced by its lower F1-Scores and Recall across all C values.
- **Polynomial Kernel**: Captures complex feature interactions effectively, providing a good balance of metrics, particularly at C=1 & C=10. However, higher C values tend to overfit the model.
- **RBF Kernel**: Matches or slightly outperforms the polynomial kernel at lower C values, demonstrating its capability to model non-linear relationships. However, it also suffers from overfitting at higher C values.

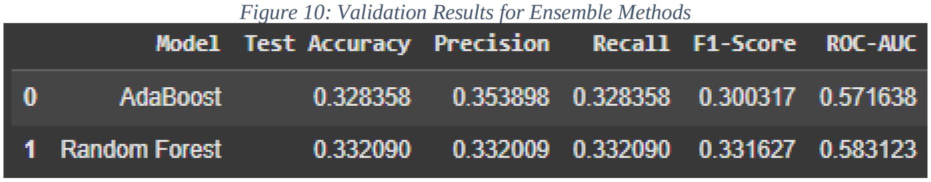
Part 4: Ensemble Methods

Ensemble methods, including AdaBoost and Random Forest, were evaluated to determine their effectiveness in improving classification performance. These methods combine multiple models to enhance predictive accuracy and robustness. The performance of each method was assessed on both the validation and test datasets the results are summarized below:



Model	Validation Accuracy	Precision	Recall	F1-Score	ROC-AUC
AdaBoost	0.358209	0.388182	0.358209	0.320618	0.588554
Random Forest	0.380597	0.400490	0.380597	0.380533	0.624024

Figure 10: Validation Results for Ensemble Methods



	Model	Test Accuracy	Precision	Recall	F1-Score	ROC-AUC
0	AdaBoost	0.328358	0.353898	0.328358	0.300317	0.571638
1	Random Forest	0.332090	0.332009	0.332090	0.331627	0.583123

Figure 11: Test Results for Ensemble Methods

From the validation and test results, **Random Forest** outperformed **AdaBoost** in most metrics. On the validation set, Random Forest achieved higher **Validation Accuracy** (0.380597) and **ROC-AUC** (0.624024), indicating better generalization and class discrimination compared to AdaBoost. On the test set, Random Forest maintained its edge, achieving higher **Test Accuracy** (0.33209) and **F1-Score** (0.33162), reflecting its ability to balance precision and recall effectively.

The superior performance of Random Forest can be attributed to its robust ensemble nature. It builds multiple decision trees and averages their predictions, reducing overfitting and improving stability. On the other hand, AdaBoost, which combines weak learners sequentially, is more sensitive to noisy data and can overfit when the dataset contains misclassified samples.

Random Forest achieved the highest test accuracy (0.332090), outperforming Logistic Regression (L2) (0.354478) and SVM (RBF kernel) (0.335821). It also balanced precision (0.332009) and recall (0.332090), making it robust against false positives and false negatives. Furthermore, it recorded the highest F1-Score (0.331627) and ROC-AUC (0.583123), demonstrating superior class discrimination and a strong balance between precision and recall.

AdaBoost performed comparably to Logistic Regression (L1), with better precision (0.353898) than Random Forest and most individual models. However, its lower recall (0.328358) and F1-Score placed it behind Random Forest in overall performance. While competitive with Logistic Regression (L2), AdaBoost fell short of Random Forest across most metrics.

Comparison with Individual Models

Logistic Regression (L2) demonstrated strong performance in Accuracy (0.354478) and Precision (0.374384), but its lower F1-Score (0.285663) and ROC-AUC (0.602323) revealed limitations in balancing Precision and Recall. In contrast, ensemble methods, especially Random Forest, offered a better overall balance across all metrics.

KNN exhibited limited performance, with its best Test Accuracy reaching 0.302239 (using Euclidean distance) and the lowest ROC-AUC (0.533224 for Euclidean). These results highlight KNN's sensitivity to distance metrics and its dependence on data distribution. Ensemble methods such as AdaBoost and Random Forest significantly outperformed KNN, underscoring the advantages of ensemble learning.

SVM with the RBF kernel achieved competitive Test Accuracy (0.335821) and ROC-AUC (0.566854). However, it lacked the consistency across metrics shown by ensemble methods. Random Forest, in particular, delivered better balance and robustness across all evaluation metrics.

Conclusion

This study highlights the performance variations of different classification algorithms on the chosen dataset. The Euclidean distance metric emerged as the most effective for KNN, achieving the highest test accuracy and F1-Score. However, Logistic Regression with L2 regularization demonstrated the best overall performance across all evaluation metrics, showcasing its ability to balance precision and recall effectively. Among the SVM kernels, the RBF kernel at C=1 provided the best overall results, with competitive accuracy and F1-Score, while the polynomial kernel offered a balanced trade-off between performance and complexity. The linear kernel, although simpler, was less effective for this dataset's complexity.

Ensemble methods proved superior to individual models, with Random Forest emerging as the most robust performer. It achieved the highest test accuracy, F1-Score, and ROC-AUC, highlighting its ability to balance precision and recall while effectively discriminating between classes. AdaBoost, though competitive in precision, lagged in recall and F1-Score compared to Random Forest. This underscores the strength of ensemble learning in leveraging multiple learners to produce reliable and generalized predictions.

Overall, the findings emphasize the importance of model selection, hyperparameter tuning, and evaluating multiple metrics to determine the most suitable approach for a given dataset. Ensemble methods, particularly Random Forest, stood out as the most effective solution for this classification task, providing robust and consistent performance across validation and test sets.

References

[1]. https://www.ibm.com/think/topics/knn

[2]. https://www.ibm.com/think/topics/logistic-regression

[3]. https://medium.com/@abhishekjainindore24/svm-kernels-and-its-type-dfc3d5f2dcd8

[4]. https://www.analyticsvidhya.com/blog/2023/01/ensemble-learning-methods-bagging-boosting-and-stacking/

[5]. https://github.com/mkjubran/ENCS5141Datasets/blob/main/ENCS5141_Exp3_MedicalCostPersonalDatasets.csv

[6]. ENCS5141 AI Lab - Experiment #3: Feature Engineering

[7]. https://scikit-learn.org/1.5/modules/neighbors.html

[8].