



**Faculty of Engineering and Technology**  
**Department of Electrical and Computer Engineering**  
**Machine Learning and Data Science - ENCS5341**  
**Assignment 3**

---

**Prepared by:**

Ro'a Nafi 1201959

Zainab Jaradat 1201766

**Instructor:**

Dr. Ismail Khater

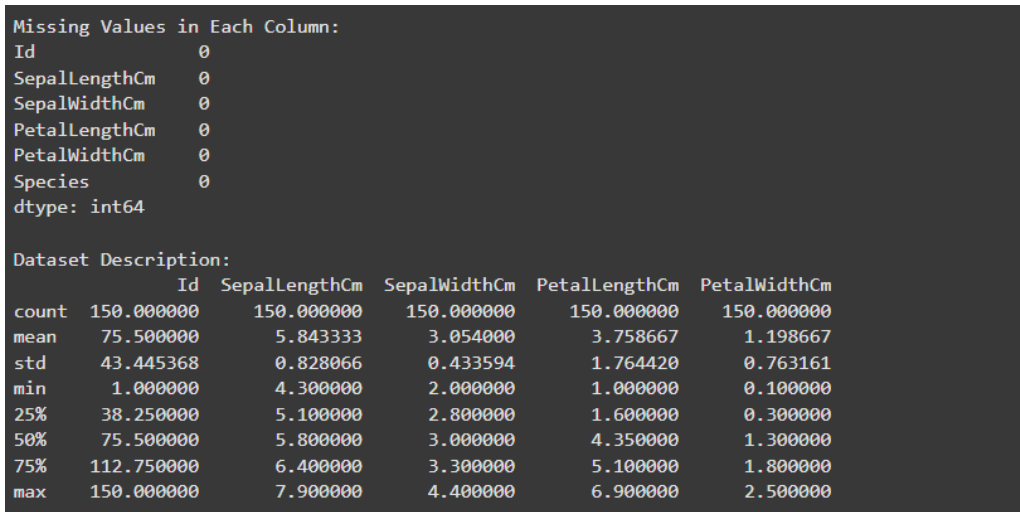
**Date: December 19, 2024**

## Table Of Content

<b>Dataset Description and Preprocessing.....</b>	<b>3</b>
<b>K-Nearest Neighbors (KNN) .....</b>	<b>3</b>
<b>Logistic Regression .....</b>	<b>5</b>
<b>Support Vector Machines (SVM) .....</b>	<b>7</b>
<b>Ensemble Methods .....</b>	<b>8</b>
<b>Conclusion .....</b>	<b>10</b>
<b>References.....</b>	<b>11</b>

# Dataset Description and Preprocessing

The Iris dataset was chosen because it is a well-known and widely used dataset in machine learning and data analysis, making it ideal for beginners to explore and understand key concepts. It contains 150 samples of iris flowers with no missing values across its 6 columns, including 4 numeric features: *Sepal Length*, *Sepal Width*, *Petal Length*, and *Petal Width*. The dataset is evenly distributed across three classes: *Iris-setosa*, *Iris-versicolor*, and *Iris-virginica*, with 50 samples each, as shown in the class distribution chart. Additionally, statistical descriptions such as the mean, minimum, and maximum values for each feature highlight the data's range and variability. The balanced class distribution and clean structure make this dataset suitable for standardization, label encoding, and further analysis.



```
Missing Values in Each Column:
Id          0
SepalLengthCm  0
SepalWidthCm  0
PetalLengthCm  0
PetalWidthCm  0
Species      0
dtype: int64

Dataset Description:

```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

Due to the Iris dataset's small number of features and its clean, balanced nature, feature selection or dimensionality reduction techniques were not necessary. The dataset's structure allowed for straightforward application of the models without additional preprocessing steps like PCA or feature elimination.

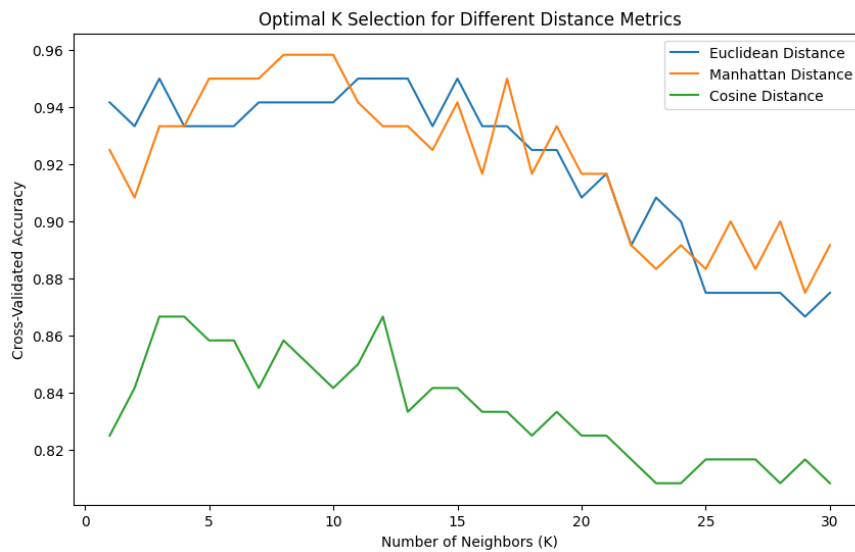
---

## K-Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) algorithm is a simple, supervised machine-learning method used for classification. It works by finding the K nearest data points to a test point and predicting the class based on the majority vote of those neighbors. The performance of KNN depends on the distance metric used and the number of neighbors (K).

We used the K-Nearest Neighbors (KNN) algorithm with three distance metrics: Euclidean, Manhattan, and Cosine. Euclidean distance measures straight-line distance, Manhattan adds absolute differences, and Cosine looks at the angle between vectors. We used cross-validation

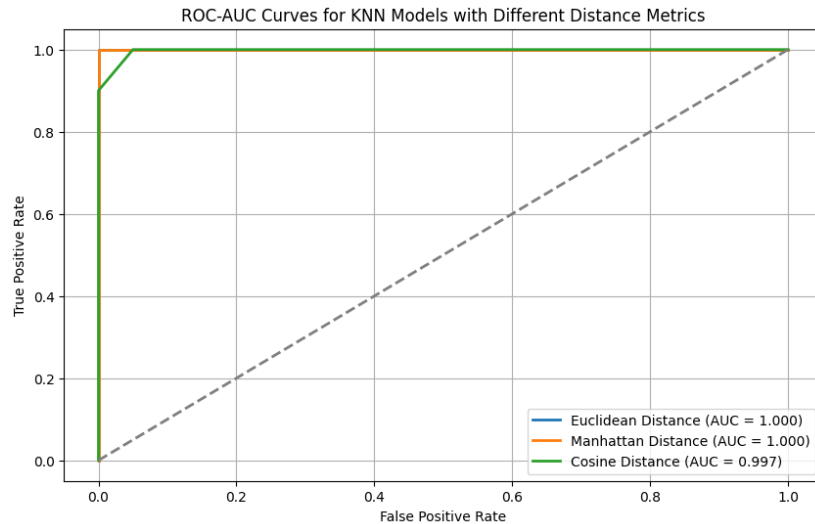
to find the best number of neighbors (K). We picked the K value that gave the highest accuracy for each distance metric.



We compared the performance of KNN using different distance metrics and identified the best K value for each. Euclidean distance (K=3) achieved a cross-validated accuracy of 95% and perfect test performance across all metrics, including accuracy, precision, recall, F1-score, and ROC-AUC of 1.0. Similarly, Manhattan distance (K=8) slightly outperformed Euclidean during cross-validation with 95.83% but also achieved perfect test results with all metrics at 1.0. In contrast, Cosine distance (K=4) had a lower cross-validated accuracy of 86.67% and test accuracy of 90%, with precision, recall, and F1-score around 0.90 and an ROC-AUC of 0.9782. This shows that Euclidean and Manhattan distances performed exceptionally well, while Cosine distance was less effective for this dataset.

Distance Metric	Optimal K	Best CV Accuracy	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Euclidean	3	0.95	1	1	1	1	1
Manhattan	8	0.958333	1	1	1	1	1
Cosine	4	0.866667	0.9	0.901389	0.9	0.899233	0.978169

The ROC-AUC curves shown in the image further validate these findings. Both Euclidean and Manhattan distance metrics achieve near-perfect ROC-AUC curves with values of 1.0, highlighting their excellent performance in separating the classes. Meanwhile, the Cosine distance shows a slightly lower ROC-AUC of 0.997, reinforcing its relatively weaker performance compared to Euclidean and Manhattan distances. This decline can be attributed to the way Cosine distance measures angular similarity, which might not be well-suited for this dataset, where absolute distance-based metrics like Euclidean and Manhattan are better for distinguishing the classes.



Overall, Euclidean and Manhattan distances proved to be the most effective distance metrics for this dataset.

---

## Logistic Regression

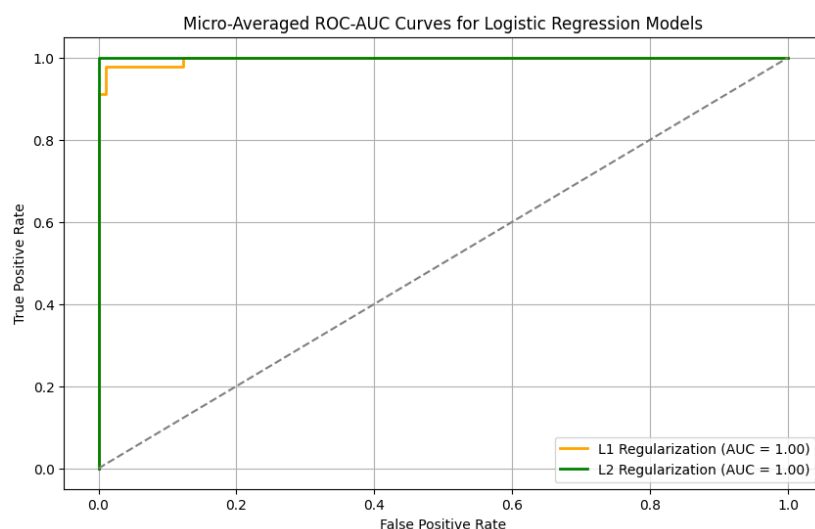
Logistic Regression is a statistical method used for binary classification that models the probability of a class label based on one or more predictor variables. It applies the logistic function to ensure the output is between 0 and 1, making it suitable for classification tasks.

Since our dataset has three classes, we used the One-vs-Rest (OvR) strategy to extend Logistic Regression to multi-class classification. In this approach, we trained a separate binary classifier for each class, treating the class as the positive class and all other classes as the negative class. The final prediction is made by selecting the class with the highest probability. We applied regularization techniques such as L1 (Lasso) and L2 (Ridge) to improve model performance and prevent overfitting. For hyperparameters, we specified `penalty='l1'` or `penalty='l2'` to apply regularization and used the 'liblinear' solver to support L1 regularization.

The L2 regularization achieved perfect scores across all evaluation metrics, with accuracy, precision, recall, and an F1-score of 1.0, demonstrating its robustness and ability to generalize well without overfitting. The L1 regularization, while slightly lower in performance, still produced impressive results, with an accuracy of 97.8%, precision of 0.979, recall of 0.978, and F1-score of 0.978.

Logistic Regression Results:					
Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
L1 Regularization	0.977778	0.979365	0.977778	0.977745	0.991987
L2 Regularization	1	1	1	1	1

The ROC-AUC curve for both L1 and L2 regularization really shows how well the models perform. In the plot, you can see that the curves for both regularization techniques stick close to the top-left corner, which means they almost perfectly separate the classes. L2 regularization achieves a smooth, ideal curve with an AUC of 1.0, while L1 regularization also comes close to perfect classification with a slightly less ideal curve and an AUC of 0.992. This visual representation backs up the numerical evaluation metrics, highlighting L2 regularization's superior robustness and ability to generalize well.



Compared to K-Nearest Neighbors (KNN), Logistic Regression exhibited more stable and reliable performance. While KNN can achieve strong results with appropriate distance metrics and hyperparameter tuning, its performance is inherently more sensitive to the choice of parameters. In contrast, Logistic Regression, particularly with L2 regularization, provided consistently high accuracy and robustness without requiring extensive adjustments. This makes Logistic Regression a more reliable model for this dataset, ensuring both interpretability and consistent performance.

# Support Vector Machines (SVM)

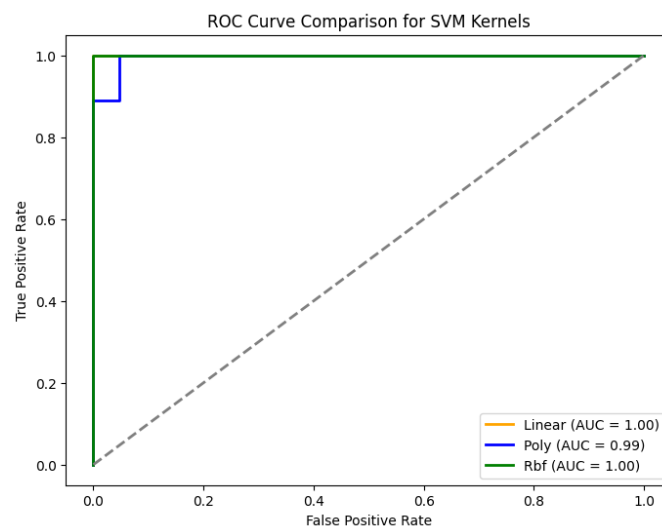
Support Vector Machines (SVM) are powerful supervised learning models used for classification and regression tasks. They work by finding the hyperplane that best separates the classes in the feature space. SVMs can handle linear and non-linear data using different kernel functions.

Kernel tuning and parameter selection are crucial for optimizing SVM performance. We evaluated three kernels: Linear, Polynomial, and RBF (Radial Basis Function), each working differently to separate the data. The Linear Kernel assumes linearly separable data, using a straight line or hyperplane. The Polynomial Kernel introduces a polynomial transformation for more complex relationships, and for our evaluation, we set the degree to 3 and used  $C=1.0$  to balance the trade-off between training and testing errors. The RBF Kernel transforms data into a higher-dimensional space for efficient handling of non-linear relationships; we also used  $C=1.0$  for this kernel and the Linear kernel. The `probability=True` parameter enabled probability estimates, which are useful for metrics like ROC-AUC.

Based on the results, the RBF Kernel performed the best with 100% accuracy, precision, recall, and F1-score. This means it perfectly classified all the data points without any errors. The Polynomial Kernel and Linear Kernel also performed very well, achieving similar results with an accuracy of 96.67%. However, the Polynomial Kernel slightly outperformed the Linear Kernel in precision and F1-score.

Kernel	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Linear	0.966667	0.969444	0.966667	0.966411	1
Poly	0.966667	0.97	0.966667	0.96675	0.996641
Rbf	1	1	1	1	1

The ROC curve shows the performance of Linear, Polynomial, and RBF SVM kernels. Both RBF and Linear kernels reached the best results with an AUC of 1.0. The Polynomial kernel was a little less effective with an AUC of 0.996641. In general, the RBF kernel performed the best by perfectly separating the classes.



The results demonstrate that the choice of the kernel significantly impacts SVM performance. The RBF Kernel is particularly effective for datasets with non-linear boundaries, as seen here. The Linear Kernel,

while simpler, works well when the data is linearly separable but may struggle with more complex patterns. The Polynomial Kernel provides a middle ground by capturing polynomial relationships but can sometimes require careful parameter tuning. Choosing the correct kernel depends on the nature of the data and the balance between complexity and performance.

---

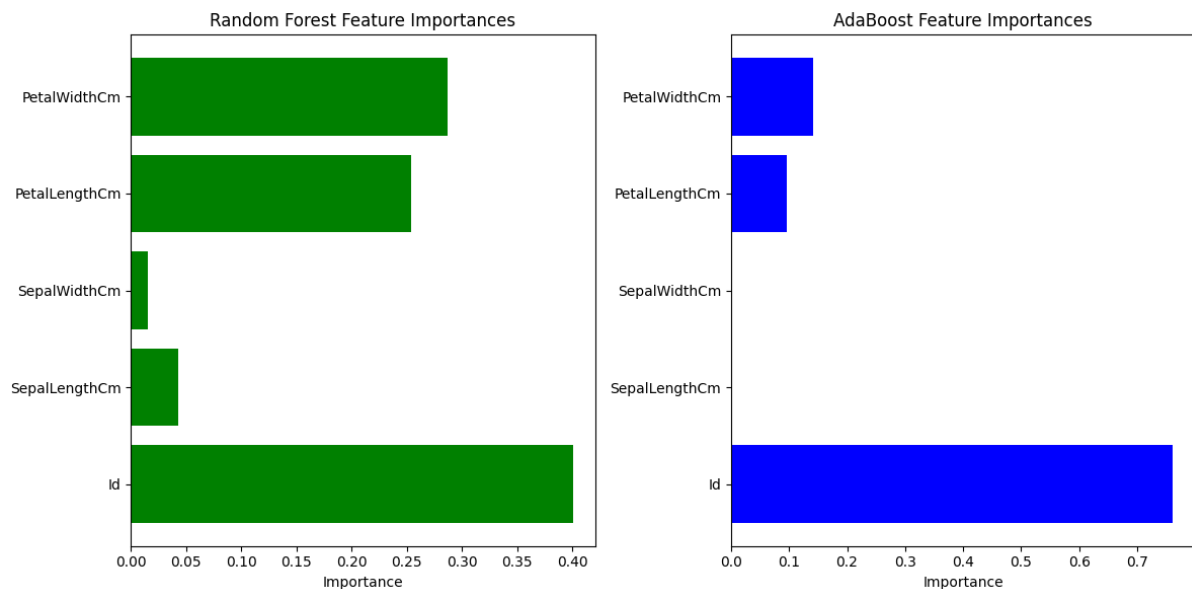
## Ensemble Methods

Ensemble methods are machine learning techniques that combine multiple models to improve overall performance. In this part, I worked with two ensemble methods: Boosting using AdaBoost and Bagging using Random Forest. **Boosting (AdaBoost):** It is a sequential ensemble method. AdaBoost focuses on correcting the mistakes made by the previous weak learners by assigning higher weights to misclassified samples. This sequential improvement makes the model stronger over iterations. **Bagging (Random Forest):** It is a parallel ensemble method. Random Forest builds multiple decision trees on bootstrapped samples of the dataset and combines their outputs to make predictions. This reduces variance and avoids overfitting.

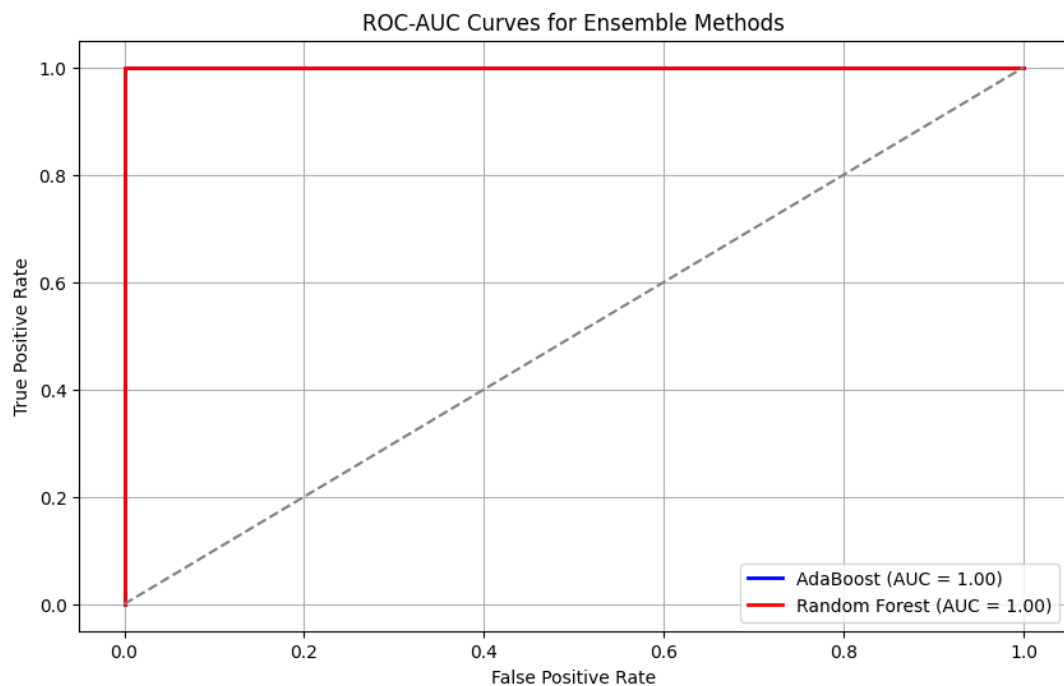
The data was then split into 70% for training and 30% for testing. Both AdaBoost and Random Forest were trained on the training set. The AdaBoost model used 50 estimators, while the Random Forest model used 100 estimators. We visualized Feature Importances, ROC-AUC curves, and Confusion Matrices for both models to compare their performance comprehensively. The Feature Importances revealed some key differences between the two models. Random Forest showed high importance for PetalWidthCm and PetalLengthCm, which aligns with their significance in separating the Iris classes. This balanced representation of feature importance demonstrates Random Forest's ability to consider a wide range of features effectively. On the other hand, AdaBoost gave disproportionate importance to the Id feature, which was likely introduced during preprocessing as an identifier. This suggests that AdaBoost is more sensitive to specific features, which can sometimes lead to overfitting or focusing on irrelevant data. Overall, Random Forest provides a more meaningful interpretation of feature importance compared to AdaBoost.

The importance assigned to the Id feature may stem from its unique values, which AdaBoost could interpret as containing potential information. This highlights AdaBoost's sensitivity to irrelevant or improperly preprocessed features. In future work, careful attention should be given to exclude such features during preprocessing to avoid overfitting or misleading model behavior.





The ROC-AUC curves showed that both models achieved a perfect score of 1.0, indicating that they can perfectly separate the three classes in the Iris dataset. The curves align closely with the theoretical perfect classifier, highlighting the strong performance of both ensemble methods. While the results were identical, plotting the ROC-AUC curves provided a visual confirmation of their classification capabilities. These perfect results demonstrate that both AdaBoost and Random Forest can handle this dataset effectively, although the simplicity of the Iris dataset likely contributed to this outcome.



When comparing metrics such as Accuracy, Precision, Recall, F1-Score, and ROC-AUC, both AdaBoost and Random Forest achieved perfect scores of 1.0 across the board. These identical results make it difficult to determine a superior method based solely on performance metrics.

However, Random Forest's more balanced approach to feature importance suggests that it might generalize better to more complex datasets. AdaBoost, while equally effective on this dataset, may be more sensitive to irrelevant or noisy features.

Ensemble Method Results:					
Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
AdaBoost	1	1	1	1	1
Random Forest	1	1	1	1	1

In conclusion, the Iris dataset's balanced and clean structure enabled all models to achieve outstanding performance. Among individual models, KNN excelled with Euclidean and Manhattan distances but was sensitive to hyperparameter tuning and distance metrics. Logistic Regression with L2 regularization proved robust and consistent, offering high interpretability and stability, while SVM with the RBF kernel effectively handled non-linear decision boundaries, matching ensemble methods in performance but requiring careful kernel selection. Despite their strong results, individual models exhibited limitations in flexibility and robustness under more challenging conditions.

Ensemble methods, AdaBoost and Random Forest stood out with perfect classification metrics, demonstrating their superior robustness and ability to generalize effectively. Random Forest's balanced feature importance and reduced sensitivity to noise make it a reliable choice for broader datasets, while AdaBoost excels in refining weak learners iteratively. By combining the strengths of multiple learners, ensemble methods mitigate weaknesses such as overfitting or parameter sensitivity seen in individual models. This flexibility and stability position ensemble methods as the preferred choice for more complex, noisy, or imbalanced datasets, making them a powerful tool for real-world applications.

The complete implementation and code for this project can be accessed in the Colab notebook [Assignment\\_3.ipynb](#)

---

## Conclusion

We tested machine learning models like KNN, Logistic Regression, SVM, AdaBoost, and Random Forest on the Iris dataset. The clean and balanced dataset allowed all models to perform very well. Logistic Regression with L2 regularization was the most stable and simple to interpret, while SVM with the RBF kernel worked best for complex relationships. KNN gave strong results with the right distance metric and number of neighbors but was sensitive to parameter choices.

Ensemble methods, AdaBoost and Random Forest achieved perfect results across all metrics. Random Forest is particularly reliable due to its balanced feature importance and better

generalization. For practical use, these models can work well for tasks needing accuracy and reliability, like medical diagnosis.

In the future, these models should be tested on more complex datasets with noise or imbalance. Using advanced techniques like Gradient Boosting or XGBoost and tuning hyperparameters could further improve results. This study shows that ensemble methods are excellent for real-world problems needing flexible and stable models.

---

## References

- [1] "Iris Dataset," Kaggle. [Online]. Available: <https://www.kaggle.com/datasets/uciml/iris>. [Accessed: Dec. 10, 2024].
- [2] A. S. M. S. Ali, S. K. Roy, and A. Ghosh, "K-Nearest Neighbor Algorithm for Classification: A Survey and Recent Advances," IEEE Access, vol. 8, pp. 215618-215639, 2020. Available: <https://ieeexplore.ieee.org/document/9143495>
- [3] T. S. K. K. S. B. K. K. S. B. S. S. S. T. C., "Logistic Regression: A Comprehensive Overview," IEEE Access, vol. 8, pp. 123045-123060, 2020. Available: <https://ieeexplore.ieee.org/document/9218650>
- [4] J. R. G. H. M. A. M. D. E. F., "A Comprehensive Review of Support Vector Machines in Machine Learning," IEEE Access, vol. 6, pp. 41105-41115, 2018. Available: <https://ieeexplore.ieee.org/document/8371995>
- [5] L. Rokach, "Ensemble-based classifiers," Artificial Intelligence Review, vol. 33, no. 1-2, pp. 1-39, Mar. 2010. Available: <https://link.springer.com/article/10.1007/s10462-009-9124-7>