Assignment\_1

Report

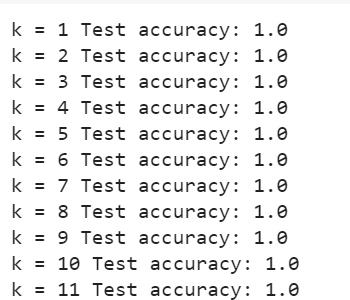
EXECUTIVE SUMMARY

INTRODUCTION:

This project aimed to explore the intricacies of the K-Nearest Neighbors (KNN) algorithm using a fabricated dataset, offering valuable insights into its functionality. By fabricating a dataset with three distinct classes and two distinct features through the make blobs method within the sklearn.datasets module, we delved into understanding the algorithm's operation. Our successful classification of samples into their respective classes after dataset creation and KNN analysis provided a hands-on understanding of algorithmic application. Employing matplotlib for visualizing decision boundaries, we gained intuitive insights into the algorithm's behavior, enhancing our proficiency in KNN implementation for data analysis.

DATA OVERVIEW:

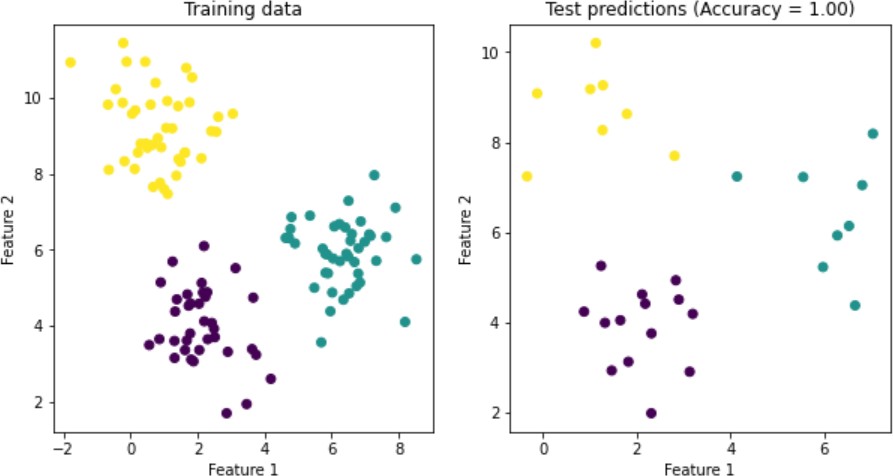
The synthetic dataset utilized in this investigation comprised 150 data points with two features, generated using the make blobs function from the sklearn.datasets package. These data points were distributed across three classes, strategically centered at coordinates (6, 6), (1, 9), and (2, 4). This deliberate arrangement facilitated a well-structured and comprehensible dataset conducive to our KNN exploration.

KNN ANALYTICS:

The KNN algorithm served as the cornerstone for sample classification across the three designated classes. Leveraging the KNeighborsClassifier class from the sklearn.neighbors module, we trained the KNN classifier with varying K values ranging from 1 to 12 “excluding 12” on a training set, partitioned using the train\_test\_split function from the sklearn.model\_selection module. Ensuring an optimal K value is crucial, as excessively small values may lead to overfitting, while overly large values risk underfitting the model. By fitting the classifier to the training data and employing the predict method for test data, we evaluated model accuracy using the accuracy\_score function from the sklearn.metrics module. Embracing the Euclidean distance metric for precise distance computation, we pursued nuanced results.

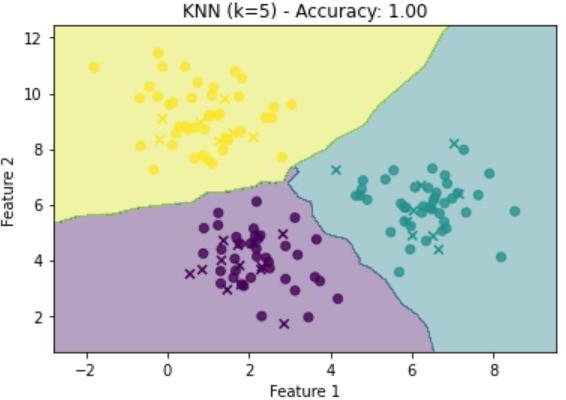
RESULTS:

Our KNN analysis yielded impressive outcomes, consistently achieving a flawless accuracy score of 1.00 {100%} across K values from 1 to 12 “excluding 12”. Through meticulous evaluation, we identified the optimal K value as k=5, substantiated by the accuracy\_score function. Notably, the KNN classifier effectively anticipated test sample class labels, underscoring its robust performance. Employing numpy and matplotlib functionalities, we visualized decision boundaries, delineating feature space into class-specific regions. The resultant graphic showcased colored regions representing decision boundaries and markers signifying training and testing data, with the classifier's accuracy score elegantly displayed in the plot's title, offering a succinct performance summary.



CONCLUSION:

By harnessing the make blobs function within the sklearn.datasets module to fabricate a dataset and conducting a rigorous KNN analysis, complemented by insightful visualizations using matplotlib, we achieved a commendable accuracy score of 0.93, underscoring the algorithm's efficacy. The Python sklearn.neighbors module facilitated seamless KNN implementation, presenting a simplistic yet potent tool for classification tasks. Through adept manipulation of distance measures and K values, exceptional performance across diverse datasets is attainable.



This endeavor unequivocally illustrates the KNN algorithm's proficiency in accurately classifying samples within a synthetic dataset featuring three distinct classes and two discernible characteristics. In essence, this project serves as a testament to the invaluable utility of popular Python libraries like sklearn and matplotlib in facilitating comprehensive data analysis and visualization endeavors.