

Abstract

The project works on a breast_cancer data-set. The aim is to compare a self-implemented K-Nearest Neighbors (KNN) algorithm with the integrated KNN classifier along with using PCA reduction to compare how data with high feature set can be reduced to achieve same results and provide better performance for processing. Starting with exploratory data analysis (EDA), the custom KNN function showed comparable accuracy, precision and recall to its built-in counterpart. In addition, principal component analysis (PCA) was used to reduce the feature space from 30 to 20 dimensions. The results showed that even with reduced dataset we were able to get same results as with a larger dataset. The main objectives of the project were to evaluate the effectiveness of a self-implemented ANN model and to investigate the impact of dimensional reduction on computational efficiency without compromising prediction accuracy in breast cancer classification.

Introduction

Aims and Objective:

The main objective of this project is a deep investigation and exploration on the application of machine learning, specifically focusing on the effectiveness of KNN algorithm in the classification of breast cancer. The project goes through three main investigations:

- Develop KNN from scratch using Manhattan distance and euclidean distance to compare the results
- Use PCA to reduce the number of features of the datasets and explore how big of a difference does the PCA make to keep the results same with using all feature set.
- Apply KNN to reduced data-set from PCA and then compare results between the Built-IN KNN, Custom KNN Function and the Using the PCA reduced dataset to evaluate using KNN

The study will aim to show the impact of performance of the model and thus contribute to the broader understanding of the efficiency of algorithms in healthcare.

Interests:

The main interest of working on this project was its direct application to healthcare. The data-set works by taking in cellular properties and then detects if someone has cancer. The breast cancer data-set comes with 30 features that all work along to decide whether someone has cancer or not and this was the most fascinating part to work on the project as these features represent different aspects of cell nuclei, this also opens a way to explore relationship between machine learning algorithms and medical diagnosis. For me personally, the project had a great appeal as this involves technology to revolutionize early cancer detection and highlights transformative impact of machine learning on healthcare.

Challenges:

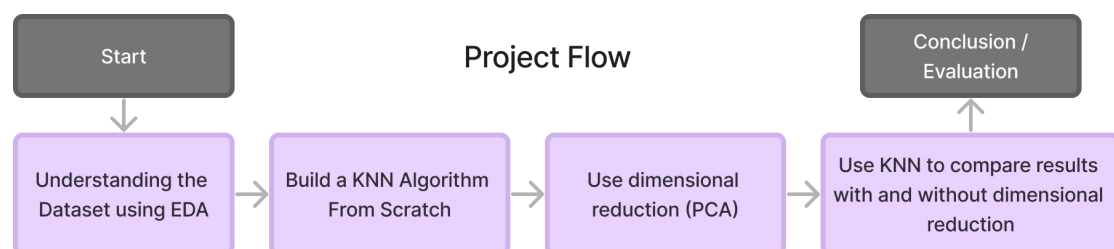
The breast cancer dataset apart from having a rich information, comes with typical challenges while analyzing data. Building a KNN from scratch poses additional challenge and requires a delicate balance between efficient algorithms and interpretability. Finally, using PCA for dimensional reduction requires a nuanced approach to understand how feature reduction affects the results of the model's ability to accurately classify cancer cases.

Significance :

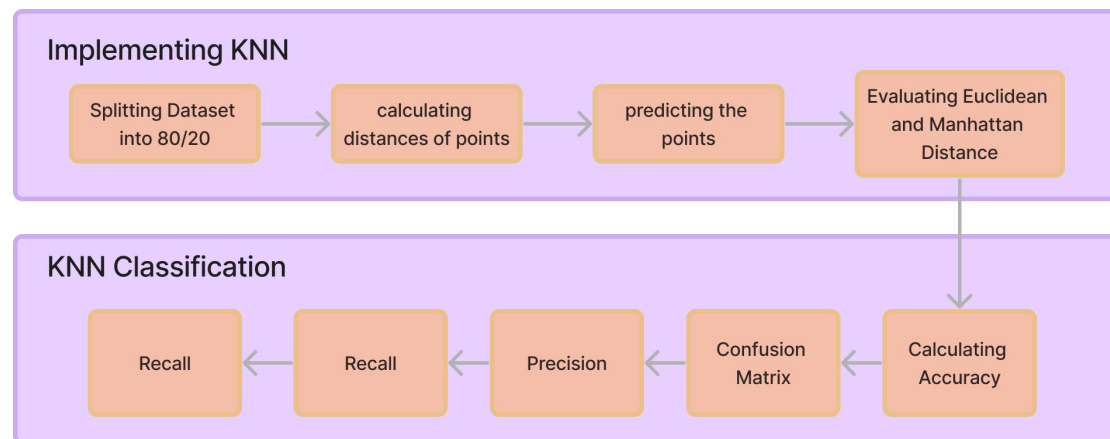
Beyond academic research, this project also serves practical significance. The involvement of machine learning and healthcare offers great opportunities for deeper and accurate diagnosis, potentially saving lives and impacting patient outcomes. The core aim of the project revolves around the understanding of performance differences between custom and integrated KNN implementations. Similarly, the impact of reducing feature dimensions contributes to academic discourse. This study also allows the practitioners and healthcare professionals about the real benefits machine learning in cancer detection.

Background

The project works on two main algorithms, KNN and PCA. Every calculation and pre-processing is done to aid the functionality and execution of these both algorithms. The diagram below explains the main structure of the project. The project starts by doing basic data analysis of the data-set. This involves inspecting the rows and columns of the data. Similarly, the analysis covers basic graphs that explain how the data set is spread. The basic data analysis allowed us to see how we can move forward with the data. As seen in the diagram there are two main algorithms used in the project. One is KNN which is coded from scratch and then the second algorithm which is PCA.



KNN



The KNN algorithm

The KNN algorithm is a simple yet powerful way of classification that is based on the proximity. As we worked on the breast cancer data-set, we explored that the breast cancer data-set only has two features, which was 0 or 1. That means that the KNN function is supposed to work by classifying the features into two sections using the technique of finding the nearest neighbour in the feature space.

Splitting Data-set into 80/20

Taking the data-set, first we separated the data-set into two parts. One part being the test data-set and the other being the predicted data-set. For our exploration, we took 80 percent of the dataset for testing and 20 percent for prediction. Then, as seen in the diagram above we needed to find the distance between the points in the data-set. This is the core aspects of implementing two main algorithms. The two algorithms that we used for the distance measure are the euclidean_distance and the Manhattan_distance. This was done to ensure that we are covering all edge aspects of implementing the KNN function and we are able to compare the results between these two approaches.

Calculating Distance of Points:

Theoretically the **Euclidean distance** is the closest path between a reference point and the second point. The euclidean distance function in the project takes two points and uses the euclidean distance mathematical formulae by taking the sum of the two points subtracted and squared and then finally by taking the square root of the answer. The algorithm uses NumPy built-in functions for taking sum and the square root. Then comes the **Manhattan distance**, this is the absolute distance between two points. Compared to the euclidean distance, in this instead of drawing a straight line to the other point we basically commute horizontally and vertically in order to calculate its distance. Now in the project the Manhattan distance function takes two points similar to the euclidean distance function. Then the function uses a for loop to iterate over the points dimensions and here we use the Python's zip lists, these allow us to combine two lists or iterable objects. Then we calculate the absolute value of the difference of the two points. This is done

using the absolute function and then we add on the absolute value till the for loop goes to the point.

Predicting using distance Functions:

Then we created two function , one for Manhattan distance and one for euclidean distance , he predict function works by iterating over the training dataset and for each value we find the distance. Now as we have two predict functions, one uses manhattan distance function that we created and the second function uses the euclidean distance. Then the function sorts the distances according to the reference point, here the function looks for the close points. Now as the main predict function takes 4 parameters and k is one of them, , here we pass the function how many closest neighbors we want to keep and therefore, we take three neighbour and keep them in this project. Then the algorithm labels the neighbour according to their class . Finally, we count the labels using binary count from pandas and return the most common. After creating these functions the project then uses the function and passes the 4 parameters that the function takes, which in our case we pass the training data-set along with telling the k value. Then we use the formulae for accuracy to find out the results of the kNN that we implemented. Now here we use the predicted results and the test data-set to compare using sum to see how different the values are and then we divide the values by the length and we get the accuracy for the KNN.

Analyze KNN function:

True Class		Function Name	Formula
Predicted Class	True Positive (TP)	Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
	False Positive (FP)	Precision	$\frac{TP}{TP + FP}$
	False Negative (FN)	Recall	$\frac{TP}{TP + FN}$
	True Negative (TN)	F1 score	$\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

After coding the KNN from scratch , it was essential to see and analyse the results for evaluation; This would allow us to see how accurate the KNN function is. Therefore , I used the Accuracy, Confusion Matrix , Precision, Recall and F1 Score to analyse the dataset.

Accuracy:

The accuracy uses the algorithm taught in the course , the algorithm basically works by using the formale for calculating accuracy. We take two parameters in the function , the test data and the predicted data that is given by our KNN function. Now, the function takes sum of the parameters and then divides it by the total number of predictions and finally returns the accuracy.

Confusion Matrix:

The confusion matrix function takes in 3 parameters , the test set, the predicted set from KNN and the number of classes in the dataset. To create a confusion matrix we start by initiating an empty matrix with the number of

classes that our dataset has , which in our case is two. Here we used the zeros function from the pandas library to populate the matrix with zeros. Then for the values passed we iterate and then we populate the true positive and the false positive in the matrix , finally we return the matrix.

Precision:

In this function we take in the confusion matrix. Firstly, we take the classes from the matrix. Then we iterate through the number of classes. Finally the core logic of the function works by implementing the precision formulae. "Precision = True positives/ (True positives + False positives)". Vedantu. (2023, December 22). Therefore , we start by finding the true positive using the values from the matrix. Then we calculate the false positive by using the false positive formulae. Here we take sum of the values from matrix and subtract them from the true positive that we calculated earlier. Finally we calculate precision, here we divide true positive from the sum of true positive and the false positive.

Recall:

In the recall function we take the confusion matrix again as a parameter, as seen in the figure above in this algorithm we use a similar algorithm as Precision , but here we take the False negative from the matrix and subtract its sum from the the true negative and then using the formulae as seen in the figure above we apply True positive divided by sum of true negative and false negative.

F1 Score:


This function takes precision and recall along with the confusion matrix as its parameters. This function also takes the number of classes from the confusion matrix. Then we iterate through the number of classes similar to the algorithms above , then we apply the formulae for F1 for every item.

Analyse Function:

This function is the main function that takes three parameters, the test dataset and the predicted values along with the target that we separated in the start. The function then basically calls all the function that we created and passes them all values. The function then converts the values reviewed into a dataframe and then concatenates all the values together in a dataframe and then returns that dataframe. This is done to ensure that there's a formal format for the way the KNN function returns the results. This also helps to make the results more readable.

PCA

To start with , I imported libraries for implementing PCA .Then the code basically takes the dataset and standardizes the data . As seen in the example figure below:



	sepal length	sepal width	petal length	petal width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

	sepal length	sepal width	petal length	petal width
0	-0.900681	1.032057	-1.341272	-1.312977
1	-1.143017	-0.124958	-1.341272	-1.312977
2	-1.385353	0.337848	-1.396138	-1.312977
3	-1.506521	0.106445	-1.284407	-1.312977
4	-1.021849	1.263460	-1.341272	-1.312977

Then , we use this dataset to find the variance of the dataset. The aim of this approach is to find out how many components would allow us to keep the 100 percent of data quality and how much components can we remove. We use pandas cumulative sum to actually plot the cumulative variance in a graph. The line graph helps us see that 20 compents have a cummulative variance of 1 . That tells us that even if we use 20 columns , we will should still get same or similar results as compared to the results we got from 30 features. Now , previously while doing exploratory data analysis , we had already observed that there were some columns that were directly correlated to each other in a heat map. That gave a hint how dimensional reduction can be used to reduce the amount of features while maintaining the quality of results.

Then after using the graph , we were able to use 20 features and therefore we created a new dataframe to store these new feature set.. Then, I separated the data again and made new variables solely for PCA. Here I moved on by using the built in KNN , this was to compare results in the end. Therefore , I imported some more libraries for implementing KNN. I used accuracy_score which was built in that allowed me to see the accuracy of the dataset that I had just created after dimensional reduction. The results were according to the expectations as they matched the results that I got with 30 features using euclidean distance in the KNN function that was built from scratch. Then I used a built in method for finding the precision , recall and f1-score. 'classification_report' allows us to input the test and predicted parameters and then it gives us a report.

Finally, the project then uses the reduced dataset along with built-in and KNN written from scratch to print the results for comparison and evaluation and to see whether the aims and objectives are accomplished.

Methodology

The project takes systematic approach to explore breast cancer dataset and implement the K-Nearest Neighbors (KNN) algorithm, both from scratch and as a built-in classifier. The project also makes use of the built-in PCA algorithm for dimensional reduction .

Exploratory Data Analysis (EDA):

The initial phase of the project involved exploring the dataset and understanding the aspects of the dataset. These included finding the features of the dataset. Looking at the target values to see how many classes did the dataset have. Similarly, we used heatmap to look at the correlating columns in the dataset. This was one of the crucial part of this project as the heatmap allowed us to understand that PCA reduction can potentially be great for this project and can help us to reduce some features in the dataset. Similarly, we were able to see that the processing for the dataset will involve two main classes.

KNN Algorithm Implementation:

The KNN algorithm was implemented from scratch. The custom implementation involved developing distance calculation methods and similarly going through the process of selecting k nearest neighbours. Similarly, we used selections to incorporate a voting mechanism for classification. Then we also used built-in KNN in order to compare results from the custom KNN with the built-in KNN. The results were similar, However, one thing that we noticed here was that the manhattan distance gave a higher accuracy and upon using the built-in kNN, the results matched the accuracy results of the euclidean distance accuracy.

Evaluation Metrics:

To measure the results of KNN models we implemented a range of evaluation metrics. These included accuracy, precision and recall. These metrics allowed us to have a comprehensive assessment of the models predictive capabilities. These evaluation metrics allowed to have insights into the algorithms strengths and the potential limitations. The choice of evaluation metrics was inspired by the fact that the project emphasized on accurate cancer diagnosis.

Principal Component Analysis (PCA):

The use of PCA dimensional reduction involved transforming the original 30 feature dataset to a reduced and smaller feature set. The implementation of this model also helped us to assess the impact of feature reduction on model performance. The number of principal components were carefully chosen to find a balance between computational efficiency and the retention of quality of dataset to maintain the results that we were able to get from the complete dataset. The reason the number was chosen to be 20 because unlike in other datasets where we can go a little lower or half the feature sets, this data aimed to have a healthcare aim, which was to identify cancer. Therefore, for such datasets it does not make sense to compromise on the quality of the data.

Comparative Analysis:

Finally, the project concludes by a comparative analysis to combine the findings and the results from the self-implemented KNN model, the built-in KNN classifier and their performance and result retention by the use of PCA dimensional reduction.

Results

Exploratory Data Analysis (EDA):

The main highlights from the basic EDA as mentioned in the methodology were that I found out that the dataset had 30 features. The heat map was an indicator of how some columns were directly correlated to each other and dimensional reduction would be the best option to reduce its features while maintaining the quality and results for the dataset. Similarly we found out the the from the target dataset 212 rows did not have breast cancer and 357 rows had cancer.

KNN Algorithm Implementation:

From the KNN that I implemented, I was able to get 93.86 percent accuracy using manhattan distance and 92.98 percent accuracy from euclidean distance.

Evaluation Metrics:

In the evaluation we used both distance methods to find out the difference in the results and upon implementation we explored that the manhattan distance had more accuracy in the results compared to the euclidean distance. Even though the difference was not big , considering this was a breast cancer detection dataset , we can see in the confusion matrix that there is a difference. In similar datasets there is almost zero tolerance for such errors or differences. Considering this dataset just had over 500 records, for a larger dataset the possibility of error can increase. However, for the sake of comparison with the built-in KNN I continued with the euclidean distance.

Euclidean Distance :

	Precision	Recall	F1-Score
0	0.781250	0.961538	0.862069
1	0.987805	0.920455	0.952941
Accuracy	0.929825		

Confusion Matrix:

		True Class	
		Positive	Negative
Predicted Class	Positive	25	1
	Negative	7	81

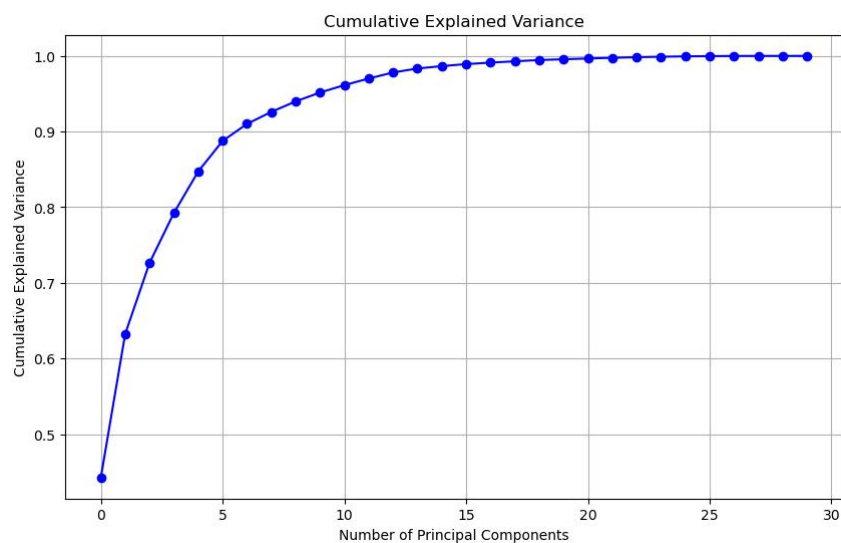
Manhattan Distance :

	Precision	Recall	F1-Score
0	0.806452	0.961538	0.877193
1	0.987952	0.931818	0.959064
Accuracy	0.938596		

Confusion Matrix:

		True Class	
		Positive	Negative
Predicted Class	Positive	25	1
	Negative	6	82

Principal Component Analysis (PCA):



The graph above shows how with increasing the components we are able to see that how much of the data quality is retained. I used this graph to evaluate my further approach and used this as a base to chose 20 components to maintain the quality and ensure that the results are not compromised in any way. Another things that was interesting in this section was that without the standardising of the dataset the graph gave me different reults which were not satisfactory. This allowed me to understand how crucial the standardising of the dataset was to ensure accurate results.

Comparative Analysis:

In the comparative Analysis we concluded by the results that using PCA and reducing the dataset from 30 to 20 did not make any change in the accuracy of the dataset.

Custom KNN with Complete Dataset of 30 Features:

	Precision	Recall	F1-Score
0	0.781250	0.961538	0.862069
1	0.987805	0.920455	0.952941
Accuracy	0.929825		

Custom KNN with PCA Reduction (20 Features)

	Precision	Recall	F1-Score
0	0.781250	0.961538	0.862069
1	0.987805	0.920455	0.952941
Accuracy	0.929825		

Evaluation

Weaknesses :

The dataset used for KNN function is comparatively simpler in terms of classes. As this dataset only had two classes. It would be more challenging for the Custom KNN function to get testing with a data that has more classes. Similarly, the dataset that was used did not have noise or null values and therefore we were able to get higher accuracy for the dataset. If we added noise to the dataset , it would increase the testing quality of the KNN function. This would also reduce the accuracy of the results. However, for a dataset that holds information that is related to health and is impactful it would be better to find out ways and approaches to increase the accuracy of the results.

Strengths:

The KNN function that is implemented from scratch proves to match the results from the built-in KNN function that proves the reliability of the algorithms used. The project also demonstrates that using manhattan distance using the custom function would actually increase the accuracy of the results. Similarly the PCA reduction technique used in the project has proved that the dataset was processed perfectly as the results equally match the results from the raw 30 feature dataset.

Conclusion:

The project works on a breast cancer dataset and explores the dataset. The project aims to write a custom KNN algorithm that would perform par with the built in KNN function and then finds out that manhattan distance provides

slightly better results. Then PCA dimensional reduction reduces the features in order to improve performance efficiency while retaining the results . The projects successfully proves that PCA reduction can help to reduce feature sets and work on a smaller dataset.

References:

- Vedantu. (2023, December 22). Precision in Math. <https://www.vedantu.com/maths/precision>
- Raihan, M. J., Khan, M. A.-M., Kee, S.-H., & Nahid, A. (2023, April 17). Detection of the chronic kidney disease using XGBoost classifier and explaining the influence of the attributes on the model using SHAP. Scientific Reports, 13, 33525. https://www.researchgate.net/figure/Confusion-matrix-and-performance-metrics-formula_fig3_370070277
- Galarnyk, M. (2022, November 29). PCA in Python Tutorial with Scikit-Learn. Built In. <https://builtin.com/machine-learning/pca-in-python>

Code References:

Manhattan Distance Implementation:

- Zach (2021, April 21). How to Calculate Manhattan Distance in Python (With Examples). Statology. <https://www.statology.org/manhattan-distance-python/>
- Piepenbreier, N. (2022, January 26). Calculate Manhattan Distance in Python (City Block Distance). Datagy. Retrieved from <https://datagy.io/manhattan-distance-python/>
- Piepenbreier, N. (2021, August 31). Python Zip Lists – Zip Two or More Lists in Python. Datagy. Retrieved from <https://datagy.io/python-zip-lists/>

Implementation of KNN Algorithm from scratch

- Bansal, I. (2022, August 3). K-Nearest Neighbors (KNN) in Python. DigitalOcean. <https://www.digitalocean.com/community/tutorials/k-nearest-neighbors-knn-in-python>
- Takahashi, K. (2016, January 6). K-Nearest Neighbor from Scratch in Python. <https://kenzotakahashi.github.io/k-nearest-neighbor-from-scratch-in-python.html>

PCA

- Galarnyk, M. (2022, November 29). PCA in Python Tutorial with Scikit-Learn. Built In. <https://builtin.com/machine-learning/pca-in-python>
- DataCamp. (n.d.). Principal Component Analysis (PCA) in Python Tutorial. DataCamp. <https://www.datacamp.com/workspace/templates/recipe-python-pca>

Data Visualization

- Kularathne, S. (2020, September 5). Prediction and Data Visualization of Breast Cancer using K-Nearest Neighbor (KNN) Classifier Algorithm. Medium. <https://medium.com/analytics-vidhya/prediction-and-data-visualization-of-breast-cancer-using-k-nearest-neighbor-knn-classifier-df7adadc4872>

KNN Implementation Using Library

- Bansal, I. (2022, August 3). K-Nearest Neighbors (KNN) in Python. DigitalOcean. <https://www.digitalocean.com/community/tutorials/k-nearest-neighbors-knn-in-python>