Date of Submission July 2024

Digital Object Identifier Not Applicable

A Study on the Implementation of K Nearest Neighbor

NIMESH GOPAL PRADHAN[1], RAMAN BHATTARAI[1]
[1]Department of Electronics and Computer Engineering, Thapathali Campus, Tribhuvan University, Kathmandu, Nepal

ABSTRACT K-Nearest Neighbors (KNN) is a simple, yet powerful ML algorithm used for classification and regression tasks. It operates by finding the K closest data points or neighbors to a given input and making predictions based on their values. KNN is non-parametric, meaning it makes no assumptions about the underlying data distribution. It is particularly useful for small datasets and scenarios where interpretability is important. However, its performance can be significantly influenced by the choice of K and the distance metric used. Feature scaling is often necessary to ensure fair contribution from all features. Despite its simplicity, KNN can achieve high accuracy and is widely used.

INDEX TERMS Classification, distance metric, neighbors, regression

I. INTRODUCTION

K-Nearest Neighbors (KNN) stands out as a fundamental and versatile algorithm in machine learning and pattern recognition, characterized by its non-parametric, instance-based approach for classification and regression tasks. The core idea behind KNN is to classify or predict the value of a data point based on the majority label or average value of its K-nearest neighbors in the feature space. Despite its simplicity, KNN is effective and widely used in various applications, from image and speech recognition to recommendation systems and medical diagnosis.

The KNN algorithm operates on the principle that similar instances are likely to be close to each other in the feature space. When a new data point is introduced, the algorithm searches for the K closest training examples in the dataset. The "distance" between data points is typically calculated using metrics such as Euclidean distance, Manhattan distance, or Minkowski distance. For classification tasks, the majority class among the K-nearest neighbors is assigned to the new data point, while for regression tasks, the prediction is usually the average of the values of the K-nearest neighbors.

One of the main advantages of KNN is its simplicity and ease of implementation. It does not require any explicit training phase or parameter estimation, making it a straightforward choice for many real-world problems. Additionally, KNN can naturally handle multi-class classification problems and adapt to various data distributions. However, the algorithm also has its limitations. As the size of the dataset grows, the computational cost of finding the nearest neighbors increases, leading to slower query times. Furthermore, KNN can be sensitive to noisy data and irrelevant features, which can affect the performance of the algorithm. To address these challenges, various enhancements and optimizations have been proposed, including dimensionality reduction techniques like PCA, distance weighting schemes, and the use of more efficient data structures such as KD-trees and Ball-trees. These improvements aim to reduce computational complexity and enhance the robustness of the KNN algorithm. KNN is also influenced by variations in the numerical values of features, which can lead to incorrect classifications. To address this, feature scaling is employed to ensure that all features contribute equally to the distance calculations.

In KNN, the choice of K, the number of nearest neighbors considered, and the method of weighting these neighbors play a crucial role in determining the performance and accuracy of the model. The parameter K controls the sensitivity of the algorithm; a small K can lead to overfitting, capturing noise and fluctuations in the data, while a large

K may smooth out distinctions between classes, leading to underfitting. The weighting schemes can further refine the algorithm's predictions by assigning different levels of influence to neighbors based on their proximity. Common approaches include uniform weighting, where all neighbors have equal impact, and distance-based weighting, where closer neighbors have more influence. Adjusting both *K* and the weighting strategy allows for fine-tuning the balance between bias and variance, enhancing the model's ability to generalize and perform effectively across various datasets.

In summary, KNN remains a popular choice in the machine learning toolkit due to its intuitive approach and versatility. Its performance and applicability in various domains make it a valuable tool for both practitioners and researchers. However, understanding its limitations and incorporating appropriate enhancements can significantly improve its effectiveness in real-world scenarios.

II. RELATED WORKS

The KNN algorithm operates on the principle that similar items are located near each other. The article [1] provides a Python implementation guide for the KNN algorithm, detailing the steps involved. Choosing the optimal value for *K* requires running the algorithm with various *K* values to find the one that minimizes errors while maintaining prediction accuracy on new data. The article also discusses the advantages and disadvantages of the KNN algorithm, including its use in recommender systems as an example of KNN-search application.

The KNN Classifier is a straightforward ML technique that classifies instances by using the nearest neighbors to determine the query's class. With improved computational power mitigating run-time performance concerns, it remains a practical choice. This paper [2] provides an overview of KNN classification, including sections on time-series similarity, retrieval speed-up, and intrinsic dimensionality, and includes Python code.

KNN is an effective classification method but suffers from low efficiency and dependency on choosing an optimal *K* value. This paper [3] presents a novel KNN-based method that replaces the original data and automatically optimizes *K* for each dataset, improving both speed and accuracy. Experiments with UCI datasets show that this new model matches the classification accuracy of C5.0 and traditional KNN while being more efficient.

KNN is a basic classification algorithm that assigns a new data point to the most frequent class among its nearest neighbors. This paper [4] presents an enhanced method combining Class Confidence Weighted (CCW) KNN with Weighted KNN (WKNN) to improve KNN performance. The optimized approach, tested on four standard datasets, showed significant accuracy gains over conventional KNN.

Despite its success, the KNN algorithm encounters challenges such as *K* computation, nearest neighbor selection, and classification rules. This paper [5] explores recent solutions to these issues, offering a roadmap for future research and new classification rules for training sample imbalance, backed by experiments on 15 UCI benchmark datasets.

III. METHODOLOGY

A. DATASET INFORMATION

The Smoking and Drinking dataset with body signal used in this study was downloaded from kaggle. The original dataset was collected from the National Health Insurance Service in Korea, contains body signal data used to predict smokers and drinkers. The dataset's primary goals are to analyze body signals and classify individuals as smokers or drinkers. It contains 991347 instances of data out of which a subset of 15000 was selected. The features in the dataset are as follows:

1) **Sex**: Gender of the individual (male or female).
2) **Age**: Age of the individual, rounded up to the nearest 5 years.
3) **Height**: Height of the individual, rounded up to the nearest 5 cm (measured in cm).
4) **Weight**: Weight of the individual (measured in kg).
5) **Sight Left**: Eyesight of the left eye.
6) **Sight Right**: Eyesight of the right eye.
7) **Hear Left**: Hearing ability of the left ear, 1 for normal, 2 for abnormal.
8) **Hear Right**: Hearing ability of the right ear, 1 for normal, 2 for abnormal.
9) **SBP**: Systolic Blood Pressure, the pressure in your arteries when your heart beats (measured in mmHg).
10) **DBP**: Diastolic Blood Pressure, the pressure in your arteries when your heart rests between beats (measured in mmHg).
11) **BLDS**: Fasting Blood Glucose level, an indicator of blood sugar levels after fasting (measured in mg/dL).
12) **Total Cholesterol**: Total cholesterol level, a measure of all the cholesterol in your blood

(measured in mg/dL).

13) **HDL Cholesterol**: HDL (High-Density Lipoprotein) cholesterol level, known as "good" cholesterol (measured in mg/dL).
14) **LDL Cholesterol**: LDL (Low-Density Lipoprotein) cholesterol level, known as "bad" cholesterol (measured in mg/dL).
15) **Triglyceride**: Triglyceride level (measured in mg/dL).
16) **Hemoglobin**: Hemoglobin level (measured in g/dL).
17) **Urine Protein**: Protein in urine, graded from 1 (negative) to 6 (positive in varying degrees).
18) **Serum Creatinine**: Serum creatinine level, which is an indicator of kidney function (measured in mg/dL).
19) **SGOT/AST**: SGOT (Serum Glutamic Oxaloacetic Transaminase) and AST (Aspartate Transaminase) levels, enzymes that help diagnose liver health (measured in IU/L).
20) **SGOT/ALT**: ALT (Alanine Transaminase) level, an enzyme that helps diagnose liver health (measured in IU/L).
21) **Gamma-GTP**: Gamma-Glutamyl Transpeptidase level, an enzyme that indicates liver and bile duct function (measured in IU/L).
22) **Smoking State**: Smoking status, 1 (never smoked), 2 (used to smoke but quit), 3 (currently smoking).
23) **Drinker**: Whether the individual drinks alcohol or not (Yes or No).

B. PREPROCESSING OF DATASET

Preprocessing the dataset is crucial to ensure effective model performance. In this study, several preprocessing steps were performed.

- **Handling Missing Values**: No missing values were found in the dataset, eliminating the need for imputation techniques.
- **Handling Categorical Features**: The categorical features were encoded using using mapping funcition of pandas.
- **Generating subset of data**: Since, the original data was very large containing 991347 instances a subset of 15000 of the dataset was taken. The subset contained 5000 instances of each target class
- **Dataset Splitting**: The subset dataset was split into training and testing sets with an 80:20 ratio. This split ensures that a majority of the data was used for training the model and a portion of the data is used to evaluate the model.

C. WORKING PRINCIPLE

K-Nearest Neighbors (KNN) is a simple, instance-based machine learning algorithm used for classification and regression. It operates on the principle that instances with similar attributes tend to belong to the same class or have similar output values. It works by finding the K closest training examples in the feature space and making predictions based on these neighbors.

In K-Nearest Neighbors (KNN), the choice of distance metric plays a pivotal role in determining the accuracy and effectiveness of the algorithm. Distance metrics measure how far apart two data points are in the feature space, guiding the algorithm in identifying the nearest neighbors for classification or regression tasks. Common metrics include Euclidean distance, which calculates the straight-line distance between points; Manhattan distance, which sums the absolute differences in each dimension; and Minkowski distance, a generalization that can be adjusted with a parameter p. Each metric can significantly impact the performance of KNN; for example, Euclidean distance works well with data where the scale of features is consistent, while Manhattan distance may be more suitable for data with high-dimensional and sparse characteristics. The choice of metric should align with the nature of the data and the specific problem at hand, as it influences how the algorithm interprets proximity and makes predictions.

Weighted K-Nearest Neighbors (Weighted KNN) extends the standard KNN algorithm by incorporating distance-based weights to enhance the predictive accuracy. In Weighted KNN, neighbors closer to the query point have a greater influence on the prediction than those farther away. This is achieved through weighting schemes such as inverse distance weighting, where weights decrease with distance, or exponential weighting, where weights drop off exponentially. By assigning more importance to nearby neighbors, Weighted KNN often produces more reliable predictions, especially in scenarios where local data points are more representative of the query point's class or value. This approach is particularly useful in handling noisy data and improving the robustness of the model.

The weights can also be applied at the feature level, allowing for a more accurate distance calculation that reflects the relative importance of each

feature. If weights are assigned to features then each feature contributes differently to the distance metric based on its assigned weight, leading to more accurate predictions by emphasizing features that are more relevant to the outcome. This method adjusts the distance formula to account for the varying significance of features, which is particularly beneficial when dealing with datasets where some features have greater predictive power or are on different scales. By incorporating feature weights, Feature-Weighted KNN improves the algorithm's ability to make precise predictions and handle complex, high-dimensional data. The significance of features can be determined through various methods that assess their relative importance in predicting the target variable. One approach is to use the domain knowledge, where insights about the problem domain guide the assignment of feature weights based on their practical relevance. Correlation analysis between features and the target variable helps in assigning higher weights to features with stronger relationships.

Using a distance metric, either with or without feature weighting, the distances between the query point and all the training points are calculated. This distance calculation is followed by neighbor selection and majority voting. The algorithm selects the k training examples with the shortest distances to the query point. These training examples are the "k-nearest neighbors" of the test example/query point. Once the k-nearest neighbors are determined, the occurrences of each class label within these neighbors are counted. The class label with the highest count is assigned to the query point.

For instance, if out of the neighbors, k_1 belong to class A, k_2 belong to class B, and k_3 belong to class C, then the query point is assigned the class label that has the highest count among these: $\text{argmax}(k_1, k_2, k_3)$.

Mathematically, the classification process can be represented as shown in Equation 1

It's important to note that the choice of k is a hyperparameter that needs to be determined. A small k can make the model sensitive to noise, while a large k can make it less sensitive to local variations. Cross-validation is often used to find the optimal value of k for a specific dataset. Similarly, while feature weighting can be beneficial, it requires careful consideration and domain knowledge to assign appropriate weights to features. Poorly chosen weights might lead to biased results or worsen the performance of the algorithm.

D. ALGORITHM OF KNN

Algorithm K-Nearest Neighbors (KNN) Classification

Input: Training dataset $D = \{(x_i, y_i)\}_{i=1}^{m}$, where x_i is a feature vector and y_i is the corresponding label; Query point x_q; Number of neighbors k; Distance metric d.
Output: Predicted label for x_q.
Step 1: Compute the distance between x_q and each training point x_i using the distance metric d.
Step 2: Sort the training points based on their computed distances from x_q.
Step 3: Select the top k training points with the smallest distances.
Step 4: Count the number of occurrences of each class label among the k nearest neighbors.
Step 5: Assign the label that appears most frequently among the k nearest neighbors to x_q.

E. DISTANCE METRICES

In K-Nearest Neighbors (KNN), various distance metrics are used to measure the similarity between data points. Each metric provides a different way of calculating distance and can significantly impact the performance of the algorithm. The following are commonly used distance metrics:

1) Euclidean Distance

Euclidean distance is a widely used metric that measures the straight-line distance between two points in the feature space. It provides a geometric interpretation where the distance represents the shortest path connecting two points. The formula for Euclidean distance is given in Equation 2. This metric is particularly effective when the features are on a similar scale and the relationships between them are approximately linear. It is well-suited for continuous data where the geometric distance directly corresponds to the meaningful difference between points.

2) Manhattan Distance

Manhattan distance, also known as L1 distance or city block distance, calculates the distance between two points by summing the absolute differences of their coordinates. This metric measures the total distance traveled along grid-like paths. The formula for Manhattan distance is shown in Equation 3. It is particularly useful in high-dimensional spaces or when dealing with sparse data. Manhattan distance tends to be less sensitive to outliers compared to Euclidean distance, making it suitable for datasets with varying scales or where feature dimensions are not equally important.

3) Minkowski Distance

Minkowski distance is a generalization of both Euclidean and Manhattan distances, parameterized

by an exponent p. It allows for flexibility in how distances are calculated based on the value of p. The formula for Minkowski distance is shown in Equation 4 For $p = 1$, Minkowski distance is equivalent to Manhattan distance, and for $p = 2$, it is equivalent to Euclidean distance. This generalization allows for adjustment based on the specific characteristics of the data.

4) Chebyshev Distance

Chebyshev distance measures the greatest absolute difference along any coordinate dimension between two points. This metric focuses on the maximum deviation across dimensions and is particularly useful in grid-like environments where movement is restricted to certain directions. The formula for Chebyshev distance is shown in Equation 5. This metric is beneficial when only the largest difference in any dimension is of interest, such as in scenarios involving constrained movement or when the feature differences are not uniformly distributed.

F. PERFORMANCE METRICS

When evaluating the effectiveness of a KNN Classifier, metrics such as Accuracy (Equation 8), Precision (Equation 9), Recall (Equation 10), and F1 Score (Equation 11) provide insights into its performance. Accuracy measures overall correctness but can be misleading with imbalanced datasets. Precision quantifies the proportion of correctly predicted positive instances among all predicted positives, while Recall assesses the proportion of correctly predicted positive instances out of all actual positives. The F1 Score balances Precision and Recall, providing a single metric that accounts for both.

G. BLOCK DIAGRAM

The system block diagram for the K-Nearest Neighbor (KNN) classification algorithm is illustrated in Figure 2. It outlines the process for designing and utilizing a KNN classifier model to predict the class of an instance. The first step involves cleaning the dataset to remove any irrelevant or noisy data and correct any errors. Then, it undergoes preprocessing, which includes converting categorical features to numerical features and scaling feature values in the dataset. Then, the dataset is split into training dataset and testing dataset. The feature weights and the number of nearest neighbors, K, are chosen for the model. The KNN model is then trained using the training dataset. Once the training is complete, the model's performance is tested using the testing dataset to evaluate how well it classifies new, unseen data. After the model has been trained and tested, various performance evaluation metrics are calculated to measure its effectiveness. The results of these performance metrics are then outputted, providing a comprehensive evaluation of the model's performance.

H. FLOWCHART

The flowchart shown in Figure 1 illustrates the sequential steps involved in constructing and analyzing a dataset using a KNN classifier. The process begins with importing the dataset. Next, the dataset is cleaned by removing irrelevant or noisy data, correcting errors, and ensuring consistency of the dataset. Categorical features are then encoded into numerical values that the model can understand. After encoding, the feature values are scaled to balance the influence of all features on the distance calculation. Once preprocessing is complete, the dataset is split into a training set and a testing set. The KNN model is trained using the training dataset and subsequently tested using the testing dataset to evaluate its performance. Performance evaluation metrics are calculated and confusion matrix is generated to assess the model's effectiveness. The results are outputted, providing a comprehensive evaluation of the model's performance. This process is repeated for different weights and values of K to obtain various KNN models. The performance metrics and confusion matrices of these models are compared to determine the optimal weights and K value for the best classification results.

IV. IMPLEMENTATION DETAILS AND RESULTS

The smoking and drinking dataset with body signal was first analyzed. It contained 991347 features which would have significantly increased the computing time so its subset of 15000 instances was taken. In the subset all the classes has 5000 instances so the class imbalance issue was not encountered. The categorical features were converted to numerical ones using the map function of pandas. The dataset can be used to classify either the smoking status or the drinking status. We choose to make the smoking status as the target class and the rest as the feature. The target class has 3 classes, 'Never Smoked', 'Used to Smoke', 'Smoking'. The subset was further divided into train and test set using a 80:20 split ratio.

A KNN classifier was instantiated with all the default parameters which was found to be as follows:

- **algorithm**: 'auto'
- **leaf_size**: 30
- **metric**: 'minkowski'
- **metric_params**: None
- **n_jobs**: -1
- **n_neighbors**: 5

- **p**: 2
- **weights**: 'uniform'

This was fitted on the training set and the prediction for the test set was obtained. The confusion matrix for this classifier is shown in Figure 3. From this we can observe the model predicted 1754 instances correctly and 1246 instances incorrectly. The classification report is also shown in Figure 4, we see the model has poor accuracy only being 0.58 and the f1-score is also not high only being 0.72.

To see the effects of feature weighting we then trained a knn classifier with all the default parameters but pass in random weights to the features. The confusion matrix for this classifier is shown in Figure 5 and the classification report is shown in Figure 6. On passing the randomly generated weights the accuracy of the model further decreased achieving only 0.56 accuracy. This shows the importance of proper weight assignment as the accuracy instead of increasing further decreased due to these weight values.

In order to properly assign weights to the feature we first found the correlation of all the features with target class. The correlation matrix is shown in Figure 28. In order to make visualization easier the bar plot of these correlation values is shown in Figure 19. The barplot for the top 5 positively correlated variables is also shown in Figure 20 From the various plots we find that the top 4 positively correlated features are 'height', 'hemoglobin', 'DRKYN', 'Weight' with values of 0.455, 0.436, 0.341 and 0.333 respectively.

A uniform weight vector containing 4 elements was generated between the value 1 and 4. A KNN classifier was trainined passing the generated weights to the top 4 correlated features and a weight of 1 to rest of the features. The confusion matrix of this classifier is shown in Figure 7 and the classification report is shown in Figure 8. The accuracy increases to 0.59 and the f1-score to 0.73. Although, still poor the accuracy is higher than the default KNN classifier. This shows how proper weight assignment can improve the accuracy of the KNN model.

We then trained KNN classifer using 'chebyshev' and 'manhattan' as the distance metrices. The confusion matrix of chebyshev KNN classifier is shown in Figure 11 and its classification report in Figure 12. This performed poorly as compared to using the 'minkowski' distance with 'p=2' i.e the 'euclidean' distance, achieving accuracy of 0.57. The confusion matrix for the manhattan KNN classifier is shown in Figure 9 and its classification report in Figure 10. This classifier performed better then the default one and has comparable result with the feature weighted classifier with proper weight assignment achieving a accuracy of 0.59.

The choice of K can also affect the performance of the model so in order to find a suitable value for K we checked in the range of 1 to 300 taking step size of 2. Furthermore we also performed K-Fold validation with fold size of 10 and compared the accuracy of each fold. This was done first by taking 'euclidean' distance metric the accuracy of K plot is shown in Figure 24. The optimal value of K is found to be 277. The confusion matrix is shown in Figure 13 and the classification report in Figure 14. The accuracy is increased to 0.62.

The above process was also done using 'manhattan' and 'chebyshev' as the distance metric. When using 'manhattan' the optimal value of K was found to be 205. Its accuracy vs K plot is in Figure 25. The confusion matrix is shown in Figure 15 and the classification report in Figure 16. The accuracy is increased to 0.62. When using 'chebyshev' the optimal value of k is found to be 67. The accuracy vs K plot is shown in Figure 26. The confusion matrix is shown in Figure 17 and the classification report in Figure 18. The accuracy is increased to 0.62.

Then we found the mutual information of the features with the target class and weighted the top 4 mutual information and trained an KNN classifier. This classifier was also passed the optimal k value which was found using cross validation to be 263 similar to above. The accuracy of K plot for this can be seen in Figure 27. The confusion matrix and classification report for this classifier can be seen in Figure 22 and Figure 23 respectively. This achieved the accuracy of 64 which is the highest of all the method tested.

From the classification report shown in figures like Figure 4, Figure 6 we can observe different metrics like accuracy, precision, recall and f1-score. There are 3 values of precision, recall, f1-score each value for each of the classes in the dataset. We can also see the number of occurrence of each class through Support and various averages(Weighted and Macro) of precision, recall and F1-score. The Macro Average is calculated for each label and calculates their unweighted mean. This does not take label imbalance into account. The Weighted Average calculates the average weighted by the support. This accounts for label imbalance.

The confusion matrix shown in figures like Figure 3, Figure 5 shows the number of accurate prediction and number of false prediction. It shows the number of True Negatives and False Positives in the first row and the number of False Negative and True Positive in the second row. True positive is the case where the model predicted the class correctly

as positive. True Negative is the case where the model predicted the class correctly as negative. False positive also known as Type 1 error, are cases where the model predicted the class as positive, but the actual class is negative and False Negative also known as Type 2 error, are cases where the model predicted the class as negative but the actual class is positive.

V. DISCUSSION AND ANALYSIS

Figure 4 and Figure 6 shows the classification report of the default KNN classifier and the KNN classifier where random weights have been used on the features. We see that on passing random wieghts the accuracy of the model decreased. This may be due to several factors, KNN relies on calculating distances between instances to determine their similarity, and these distances are significantly influenced by the feature values and their respective weights. When random weights are introduced, they can distort these distances in an unpredictable manner. This distortion can disrupt the model's ability to accurately identify the nearest neighbors, leading to less effective classification. Moreover, random weights may introduce noise into the distance calculations, making it harder for the model to generalize from the training data to unseen examples. The random assignment of weights may sometime increase the accuracy if the random weights align in a way that enhances the distance calculation and neighbor selection process and unintentionally improve the model's performance. But the weights assignment should not be left to chance and should be done properly.

Figure 8 shows the classification report of KNN classifier where the top 4 positively correlated features have been assigned weights between 1 and 4 and the rest of the features are assigned 1 as their weight. This model performed better than the default one this is because the properly weighted KNN classifier better prioritized features that have a stronger relationship with the target variable, allowing it to make more informed decisions during the classification process. By assigning higher weights to these important features, the model was able to capture more relevant information and reduce the impact of less informative features. This approach not only improved the model's ability to discriminate between classes but also made the classifier more sensitive to features that are crucial for accurate predictions. This demonstrates that proper weighting of features based on correlation can lead to better model performance compared to using random weights. Assigning weights based on the top 4 positively correlated features did result in a slight improvement in accuracy from 0.58 to 0.59. However, the increase in accuracy was modest due to several reasons. The features selected for weighting, despite having positive correlations with the target variable, might not capture the full complexity of the relationships within the dataset. Additionally, KNN's performance is sensitive to the weighting of features, but if the remaining features or their interactions are still influential, the model's overall accuracy might not improve substantially. The modest gain also reflects that feature weighting is just one of many factors influencing model performance. Other aspects such as feature scaling, data quality, and the choice of distance metric can also play critical roles.

When changing the distance metric to 'Chebyshev' it got the lowest accuracy of 0.56. The Chebyshev distance, which measures the maximum absolute difference between coordinates, can be less effective in capturing the nuances of feature relationships in datasets where the scale or distribution of features varies significantly. This distance metric is sensitive to the largest individual differences, which may not always align well with the underlying patterns needed for classification. As a result, Chebyshev distance might not reflect the true similarities between instances as effectively as the Euclidean or Manhattan distances, leading to decreased accuracy. On the other hand, when using the Manhattan distance the accuracy was highest at 0.59. The Manhattan distance, which calculates the sum of absolute differences between coordinates, often performs better in high-dimensional spaces where feature scales are different or when there are outliers. It is less affected by extreme values compared to Euclidean distance, which considers the square of differences. The Manhattan distance's robustness to feature scales and its ability to handle outliers make it more suitable for some datasets,which may be why it achieved higher accuracy than both Chebyshev and Euclidean distances. The fluctuation in accuracy among these metrics, although notable, was not dramatic. This modest variation suggests that while the choice of distance metric does impact model performance, the differences might be relatively minor depending on the dataset. This outcome shows the importance of considering various metrics in KNN and highlights that while selecting an optimal distance metric can enhance performance, other factors such as feature scaling, data quality, and model hyperparameters also play significant roles.

On the search of the optimal value of K using different distance metric we found the optimal value to be different for different distance metric. For euclidean it was 277, for chebyshev it was 67 and for manhattan it was 205. The optimal K value

varies for different distance metrics because each metric measures similarity between data points in distinct ways. The Euclidean distance captures the straight-line distance between points, the Chebyshev distance measures the maximum coordinate difference, and the Manhattan distance sums the absolute differences between coordinates. These different approaches to measuring distance can lead to variations in how the nearest neighbors are identified, influencing the optimal K value for achieving the best classification performance. Using the optimal K value for different distance metric improved accuracy in all cases because it represents the best trade-off between bias and variance for the specific distance metric. A well-chosen K value can enhance the model's ability to generalize from the training data to unseen data, reducing overfitting or underfitting. This optimization process helps in identifying the number of neighbors that provide the most reliable classification results.

When using the optimal K values, the choice of distance metric had less impact, as all metrics achieved an accuracy of 0.62. This convergence suggests that the selection of K was a more critical factor than the choice of distance metric when each is optimized for our dataset. At the optimal K, each metric is able to effectively capture the underlying structure of the data, leading to similar performance levels. This highlights the importance of tuning hyperparameters to achieve the best results, rather than relying solely on the inherent properties of the distance metrics.

The accuracy vs. K graph (Figure 24, Figure 25, Figure 26) showed lower accuracy for smaller K values (e.g., 1 to 15) because small K values can lead to overfitting. With fewer neighbors, the model is more sensitive to noise and specificities of the training data, which reduces its ability to generalize. As K increases, the model averages over more neighbors, which generally leads to more stable and reliable predictions, hence slightly higher accuracy. However, beyond a certain point, increasing K can lead to underfitting, where the model becomes too generalized and loses its ability to capture important distinctions in the data. The balance point, where accuracy stabilizes, indicates the optimal trade-off for the chosen distance metric and dataset. The accuracy capping at 0.62, despite optimizing K, indicates that there might be inherent limitations in the dataset. Extending the search range for K beyond 1 to 301 might not necessarily yield better results if the underlying data distribution and feature set cannot support higher accuracy. It could also suggest that other factors, such as feature selection, feature scaling, or even the inherent noise in the data, are limiting the model's performance.

The difference between the top features identified by correlation and mutual information arise due to the fundamental differences between these two measures. Correlation measures the linear relationship between two variables, assessing how one variable changes with respect to another. Mutual information, on the other hand, captures any kind of dependency between variables, not just linear. This means mutual information can detect more complex, non-linear relationships that correlation might miss. This might also be the reason why the KNN classifier where the top 4 mutual information feature are weighted performed the best with accuracy of 0.64 whereas the classifier where correlation was used as the feature selection criteria the highest accuracy was only 0.62.

VI. CONCLUSION

This study successfully applied K-Nearest Neighbors (KNN) analysis to a smoking and drinking dataset to identify patterns and relationships within the data. We implemented various KNN classifiers, including the default KNN, weighted KNN, and KNN with different parameter modifications, to construct a model that accurately classified instances based on the provided features. The model's performance, evaluated using metrics like accuracy, precision, recall, and F1-score, demonstrated its effectiveness in predicting outcomes.

Evaluating these classifiers provided crucial insights into optimizing performance. Initially, the default KNN classifier showed moderate accuracy and F1-score. Efforts to improve the model with randomly assigned feature weights resulted in decreased accuracy, highlighting the necessity of proper weight assignment. However, weighting features based on their correlation with the target class enhanced both accuracy and F1-score. Additionally, testing different distance metrics revealed that Chebyshev distance performed poorly, while Manhattan distance achieved comparable results to the default classifier. Optimizing the number of neighbors K further improved accuracy across various distance metrics.

Despite these enhancements, several limitations were identified, including computational complexity with increasing K values, sensitivity to hyperparameters, the non-parametric nature of the KNN model, performance degradation in higher-dimensional spaces, and sensitivity to outliers and noise. Future improvements could explore ensemble methods to enhance the model's predictive power and robustness. Overall, this paper provides valuable insights into machine learning, laying a foundation for future research and development of similar models in different domains.

APPENDIX

A. EQUATIONS

1) KNN Classification Rule

$$y_{\text{pred}} = \text{argmax}_y \sum_{i=1}^{k} 1(y_{\text{train},i} = y) \tag{1}$$

where: where y_{pred} is the predicted class label, $y_{\text{train},i}$ is the class label of the i-th nearest neighbor, and $1(\cdot)$ is the indicator function.

- y_{pred} is the predicted class label.
- $y_{\text{train},i}$ is the class label of the i-th nearest neighbor.
- $1(\cdot)$ is the indicator function.

2) Euclidean Distance

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2}$$

where $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ are the coordinates of the two points in an n-dimensional space.

3) Manhattan Distance

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i| \tag{3}$$

where $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$.

4) Minkowski Distance

$$d(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} \tag{4}$$

where $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$.

5) Chebyshev Distance

$$d(x,y) = \max_{i=1}^{n} |x_i - y_i| \tag{5}$$

where $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$.

6) Pearson Correlation Coefficient

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2 \sum(Y_i - \bar{Y})^2}} \tag{6}$$

where:

- X_i are the individual values of the first variable.
- Y_i are the individual values of the second variable.
- $\bar{X}$ is the mean of the first variable.
- $\bar{Y}$ is the mean of the second variable.

7) Mutual Information

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{7}$$

where:

- X and Y are the discrete random variables.
- $p(x,y)$ is the joint probability distribution function of X and Y.
- $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y, respectively.
- $I(X;Y)$ is the mutual information, measuring the amount of information obtained about one random variable through the other random variable.

8) Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

where:

- TP (True Positives) are the correctly predicted positive instances.
- TN (True Negatives) are the correctly predicted negative instances.
- FP (False Positives) are the incorrectly predicted positive instances.
- FN (False Negatives) are the incorrectly predicted negative instances.

9) Precision

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

10) Recall

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

11) F1 Score

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

B. FIGURES

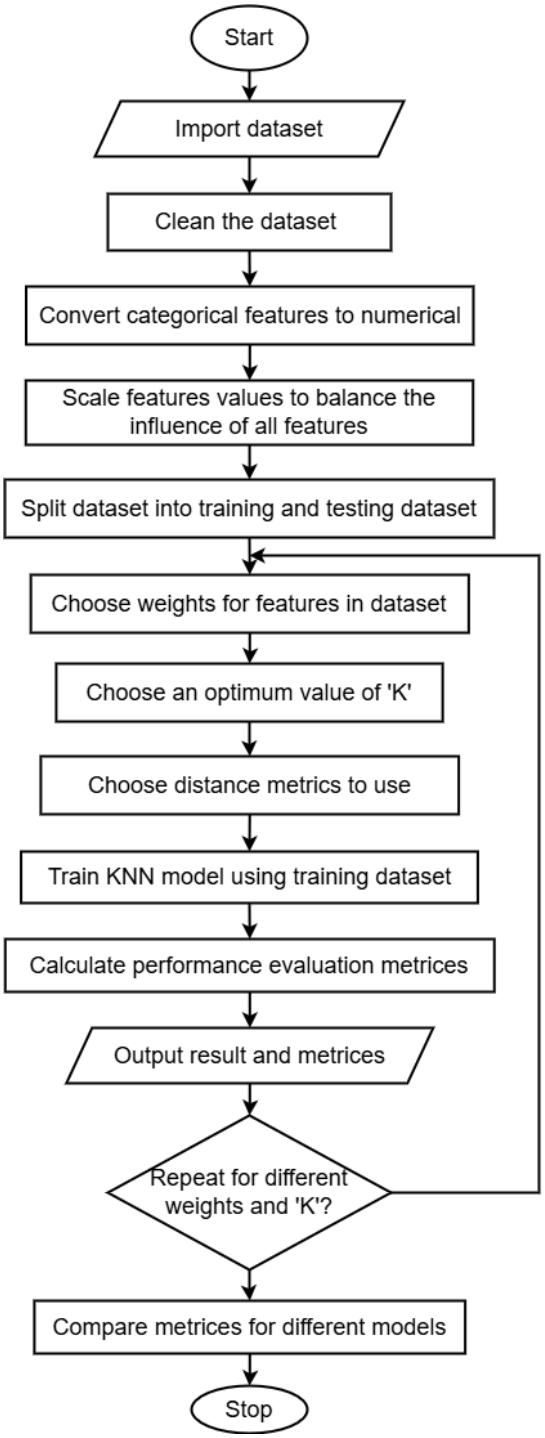

FIGURE 1. Flowchart

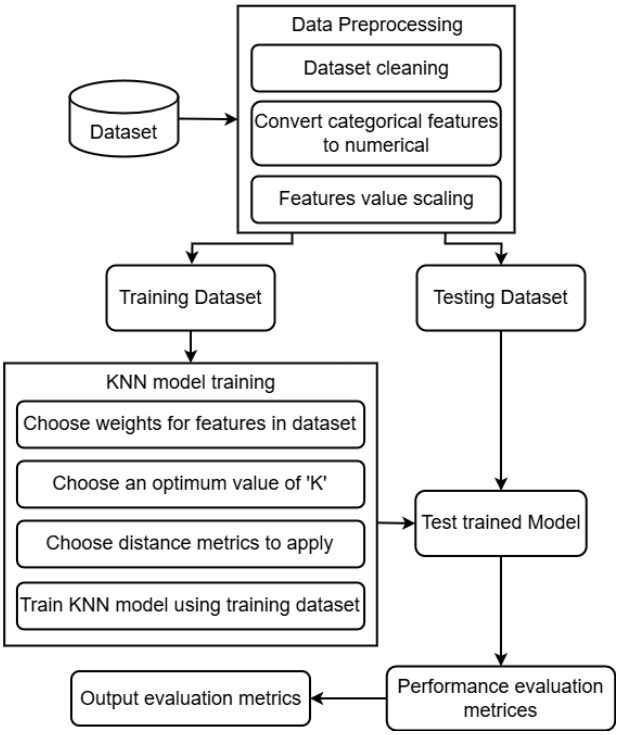

FIGURE 2. System Block Diagram

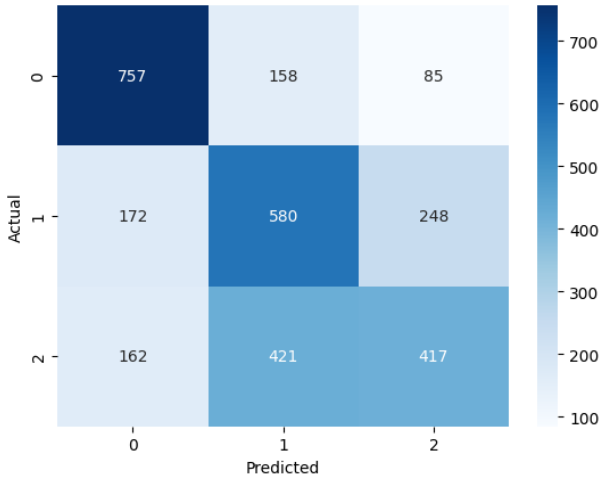

FIGURE 3. Confusion matrix of default KNN classifier

	precision	recall	f1-score	support
1.0	0.69	0.76	0.72	1000
2.0	0.50	0.58	0.54	1000
3.0	0.56	0.42	0.48	1000
accuracy			0.58	3000
macro avg	0.58	0.58	0.58	3000
weighted avg	0.58	0.58	0.58	3000

FIGURE 4. Classification report of default KNN classifier

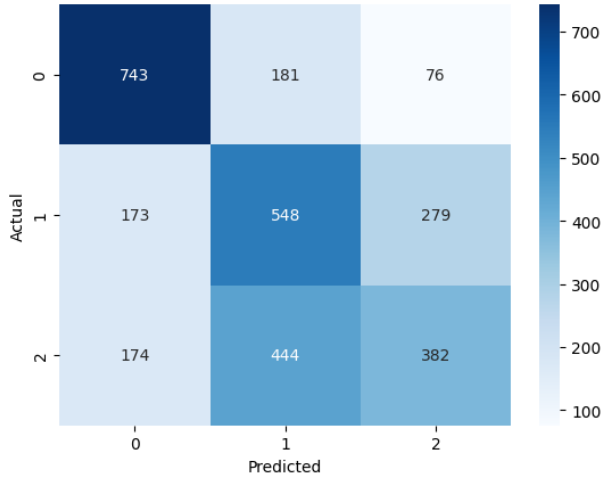

FIGURE 5. Confusion matrix of random feature weighted classifier

	precision	recall	f1-score	support
1.0	0.68	0.74	0.71	1000
2.0	0.47	0.55	0.50	1000
3.0	0.52	0.38	0.44	1000
accuracy			0.56	3000
macro avg	0.56	0.56	0.55	3000
weighted avg	0.56	0.56	0.55	3000

FIGURE 6. Classification report of random feature weighted classifier

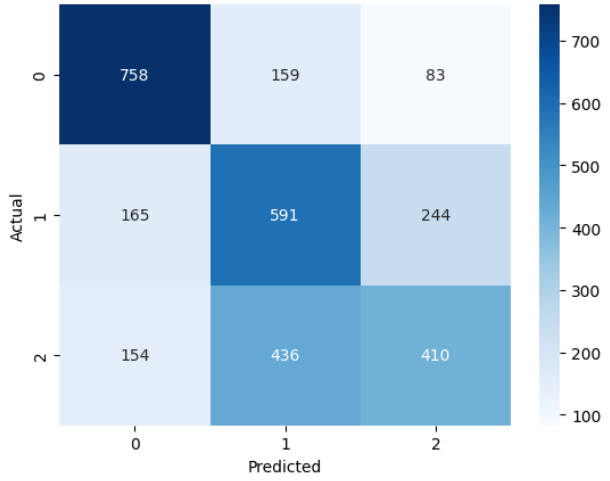

FIGURE 7. Confusion matrix of top 4 positively correlated features weighted

	precision	recall	f1-score	support
1.0	0.70	0.76	0.73	1000
2.0	0.50	0.59	0.54	1000
3.0	0.56	0.41	0.47	1000
accuracy			0.59	3000
macro avg	0.59	0.59	0.58	3000
weighted avg	0.59	0.59	0.58	3000

FIGURE 8. Classification report of top 4 positively correlated features weighted

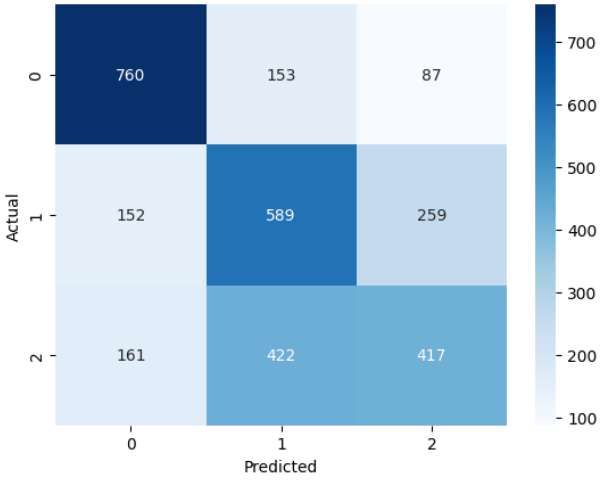

FIGURE 9. Confusion matrix using Manhattan distance metric

	precision	recall	f1-score	support
1.0	0.71	0.76	0.73	1000
2.0	0.51	0.59	0.54	1000
3.0	0.55	0.42	0.47	1000
accuracy			0.59	3000
macro avg	0.59	0.59	0.58	3000
weighted avg	0.59	0.59	0.58	3000

FIGURE 10. Classification report using Manhattan distance metric

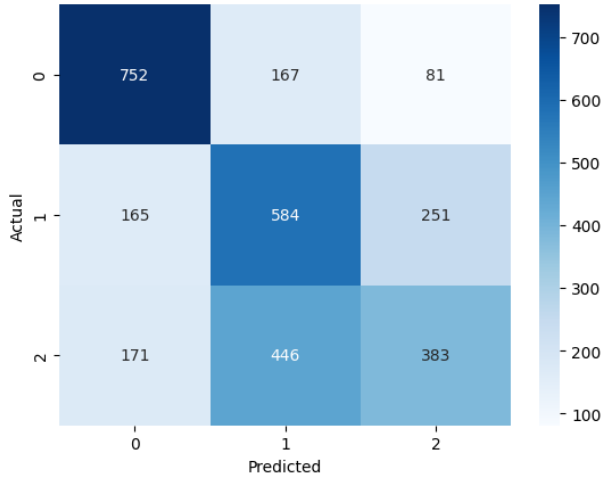

FIGURE 11. Confusion matrix using Chebyshev distance metric

	precision	recall	f1-score	support
1.0	0.69	0.75	0.72	1000
2.0	0.49	0.58	0.53	1000
3.0	0.54	0.38	0.45	1000
accuracy			0.57	3000
macro avg	0.57	0.57	0.57	3000
weighted avg	0.57	0.57	0.57	3000

FIGURE 12. Classification report using Chebyshev distance metric

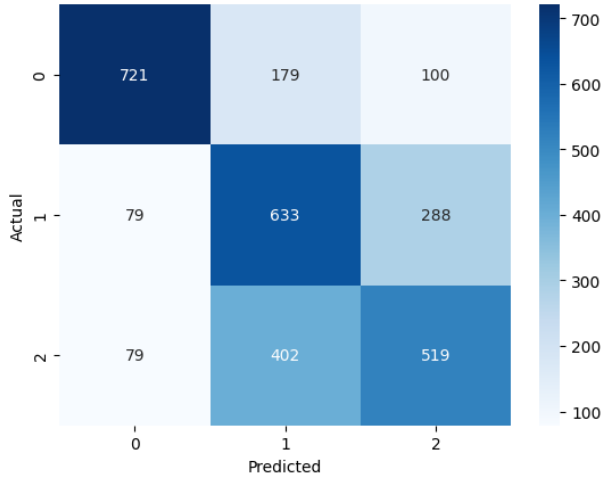

FIGURE 13. Confusion matrix using euclidean distance metric and optimal K neighbor

	precision	recall	f1-score	support
1.0	0.82	0.72	0.77	1000
2.0	0.52	0.63	0.57	1000
3.0	0.57	0.52	0.54	1000
accuracy			0.62	3000
macro avg	0.64	0.62	0.63	3000
weighted avg	0.64	0.62	0.63	3000

FIGURE 14. Classification report using euclidean distance metric and optimal k neighbor

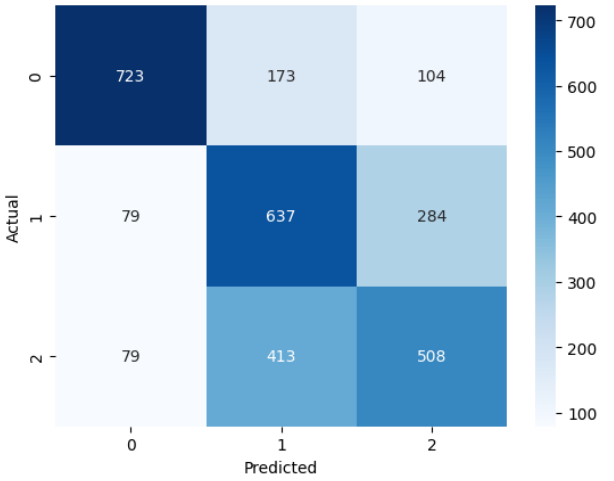

FIGURE 15. Confusion matrix using Manhattan distance metric and optimal K neighbor

	precision	recall	f1-score	support
1.0	0.82	0.72	0.77	1000
2.0	0.52	0.64	0.57	1000
3.0	0.57	0.51	0.54	1000
accuracy			0.62	3000
macro avg	0.64	0.62	0.63	3000
weighted avg	0.64	0.62	0.63	3000

FIGURE 16. Classification report using Manhattan distance metric and optimal k neighbor

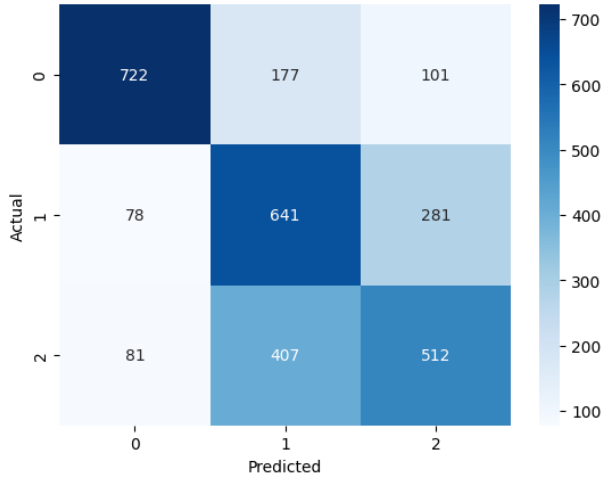

FIGURE 17. Confusion matrix using Chebyshev distance metric and optimal K neighbor

	precision	recall	f1-score	support
1.0	0.82	0.72	0.77	1000
2.0	0.52	0.64	0.58	1000
3.0	0.57	0.51	0.54	1000
accuracy			0.62	3000
macro avg	0.64	0.62	0.63	3000
weighted avg	0.64	0.62	0.63	3000

FIGURE 18. Classification report using Chebyshev distance metric and optimal k neighbor

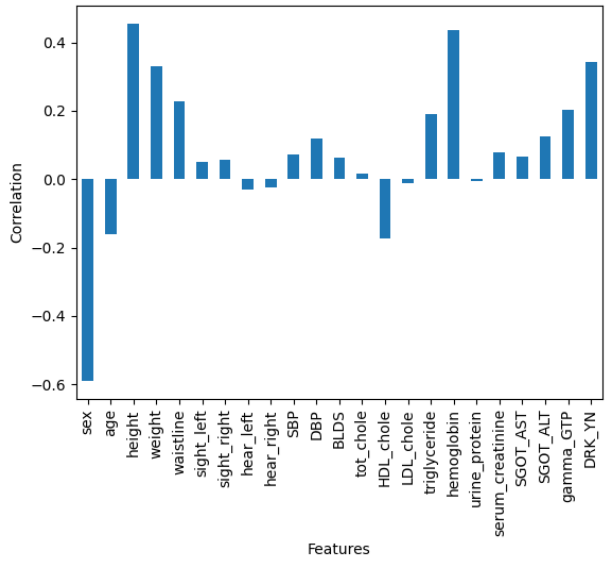

FIGURE 19. Correlation bar plot

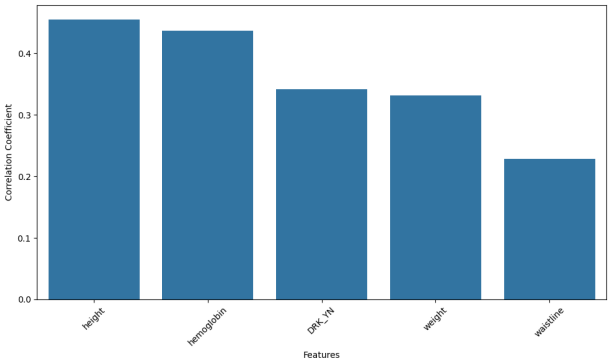

FIGURE 20. Bar plot of top 5 positively correlated features

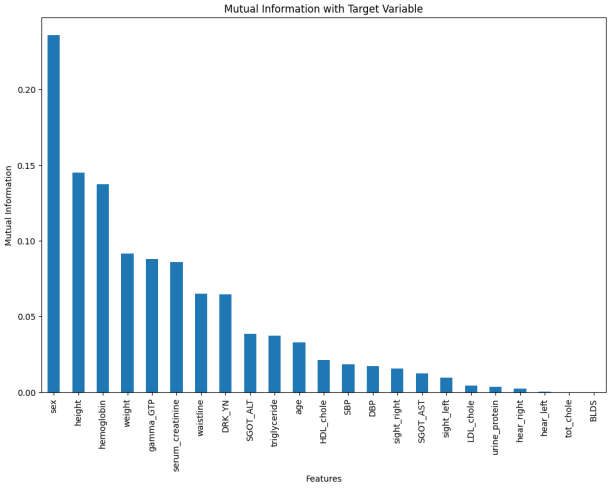

FIGURE 21. Bar plot of mutual information

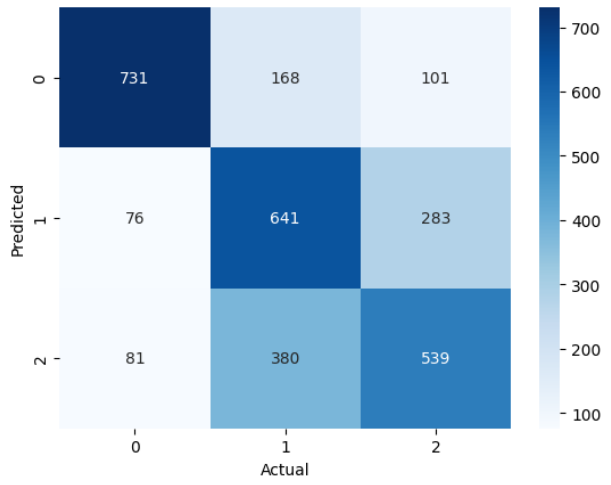

FIGURE 22. Confusion Matrix using Manhattan distance and optimal K neighbor where top 4 features are weighted based on mutual information

	precision	recall	f1-score	support
1.0	0.82	0.73	0.77	1000
2.0	0.54	0.64	0.59	1000
3.0	0.58	0.54	0.56	1000
accuracy			0.64	3000
macro avg	0.65	0.64	0.64	3000
weighted avg	0.65	0.64	0.64	3000

FIGURE 23. Classification report using Manhattan distance and optimal K neighbor where top 4 features are weighted based on mutual information

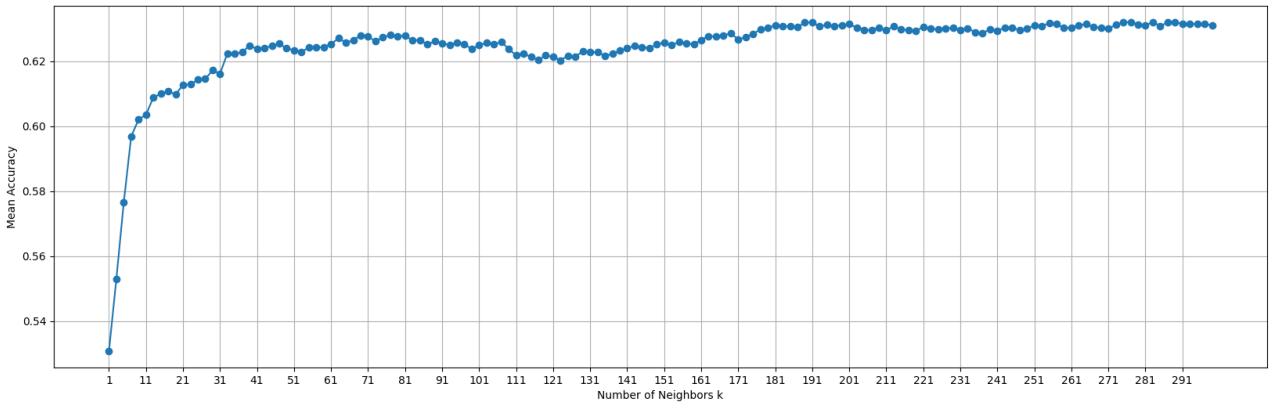

FIGURE 24. **Accuracy vs K using euclidean distance metric**

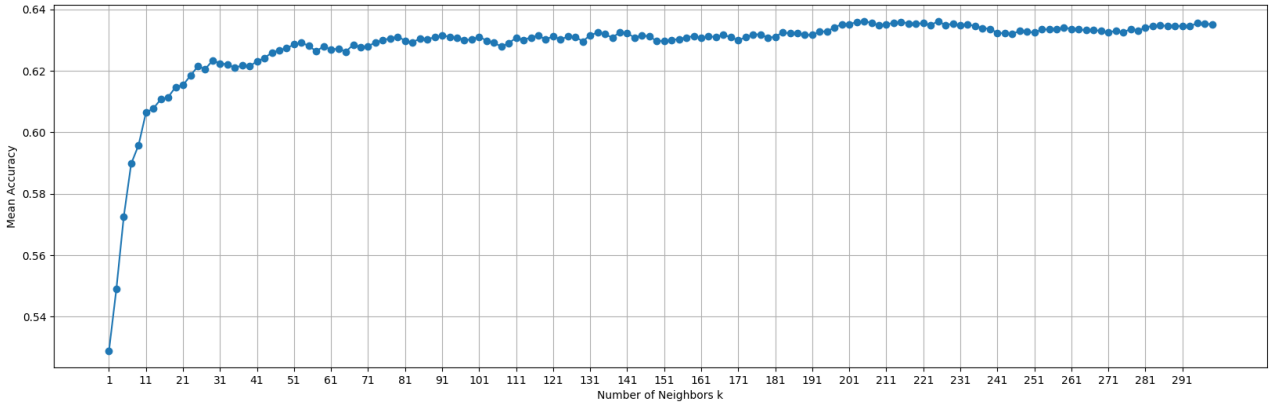

FIGURE 25. **Accuracy vs K using Manhattan distance metric**

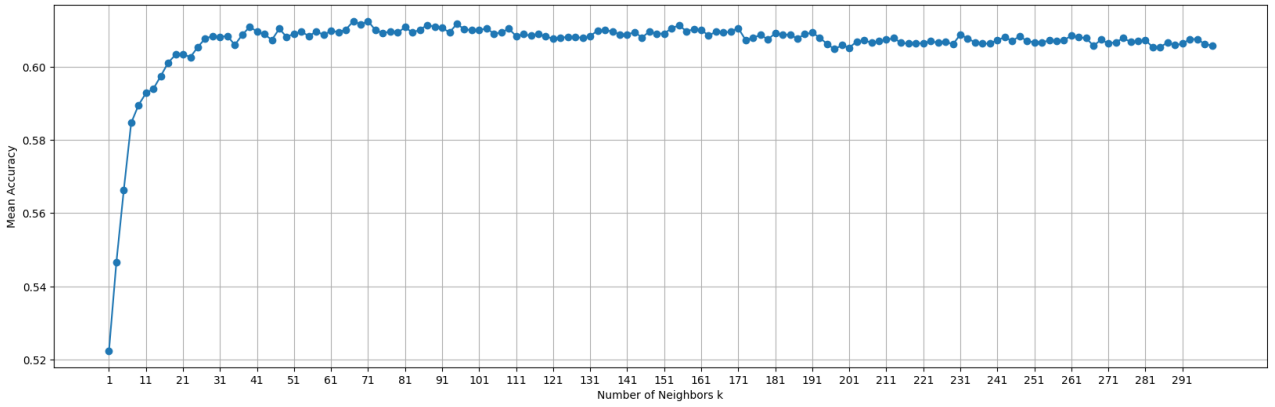

FIGURE 26. **Accuracy vs K using Chebyshev distance metric**

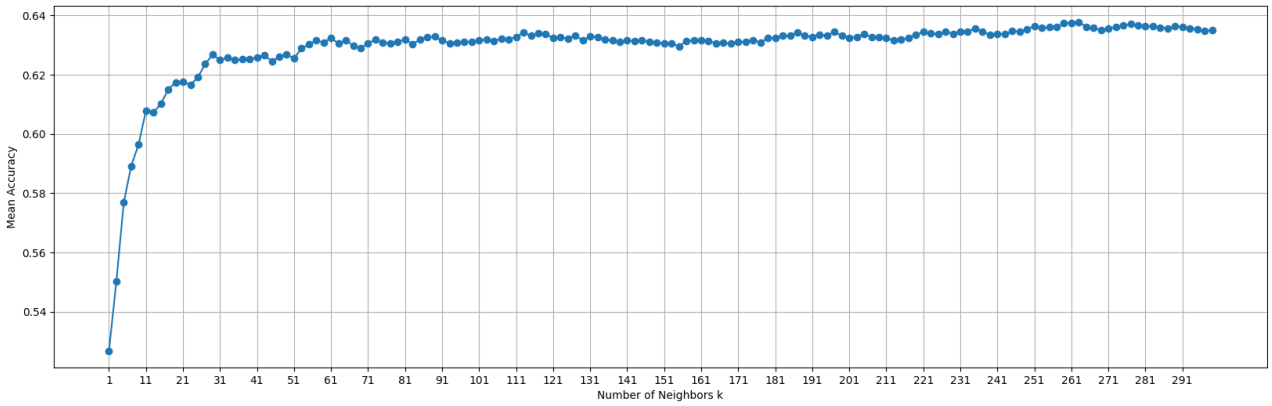

FIGURE 27. Accuracy vs K using Manhattan distance metric where top 4 features are weighted based on mutual information

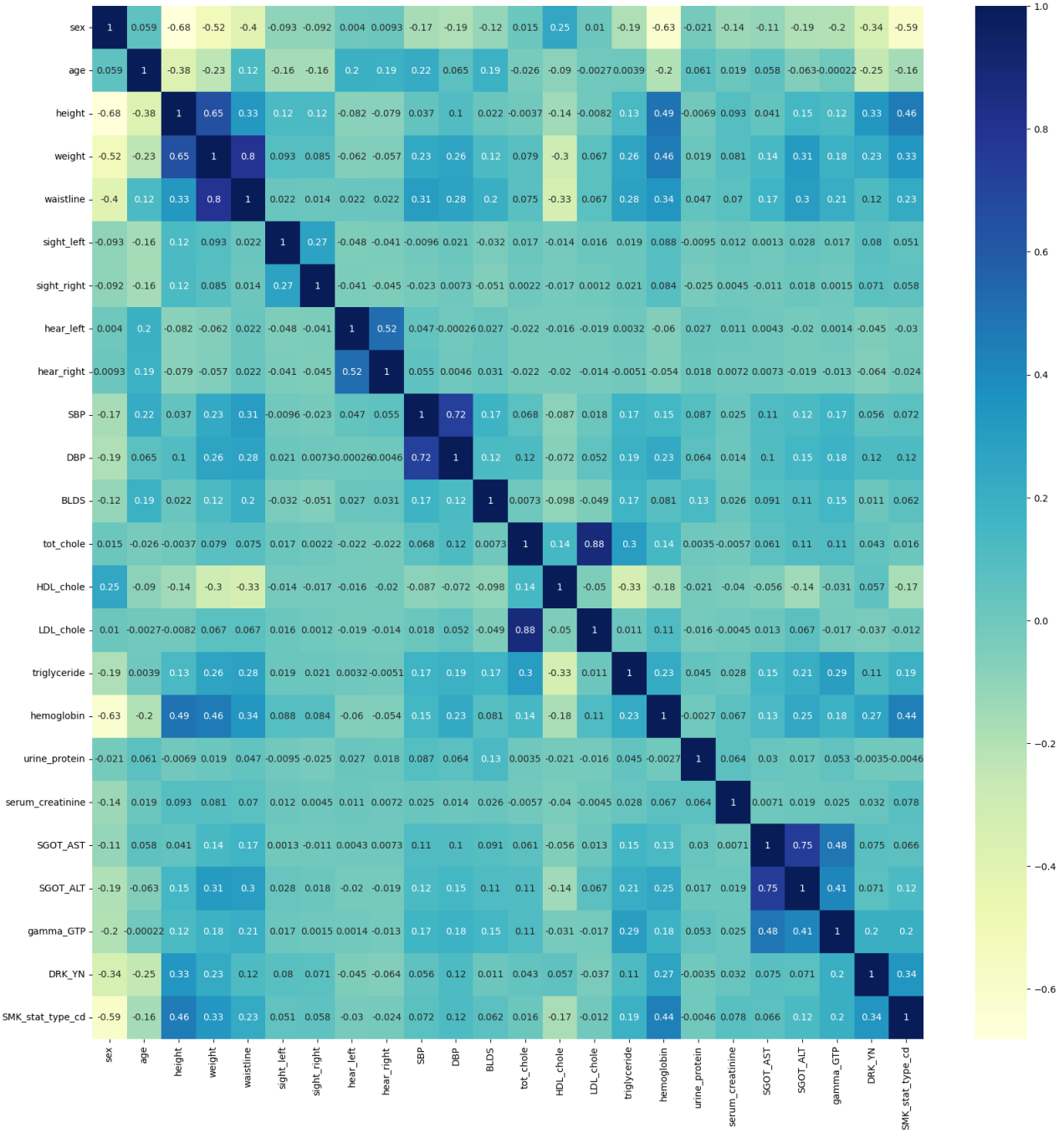

FIGURE 28. Correlation Matrix

REFERENCES

[1] O. Harrison, "Machine Learning Basics with the K-Nearest Neighbors Algorithm," *Towards Data Science*, Sep. 11, 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. [Accessed: Jul. 21, 2024].

[2] P. Cunningham and S. Delany, *k-Nearest Neighbour Classifiers – 2nd Edition*. 2020.

[3] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN Model-Based Approach in Classification," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, R. Meersman, Z. Tari, and D.C. Schmidt, Eds. Berlin, Heidelberg: Springer, 2003, vol. 2888, pp. 511–524.

[4] K. Joshi, S. Jain, S. Kumar, and N. Roy, "Optimization of K-Nearest Neighbors for Classification," in *Futuristic Trends in Network and Communication Technologies*, P.K. Singh, G. Veselov, V. Vyatkin, A. Pljonkin, J.M. Dodero, and Y. Kumar, Eds. Singapore: Springer, 2021, vol. 1395, Communications in Computer and Information Science.

[5] S. Zhang, "Challenges in KNN Classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 10, pp. 4663-4675, Oct. 2022.

NIMESH G. PRADHAN is currently pursuing a Bachelor's degree in Electronics, Communication, and Information Engineering at Thapathali Campus. He is currently in the final year of his degree. His interests lie in the fields of Data Mining, Machine Learning, and Deep Learning.

RAMAN BHATTARAI is currently pursuing a Bachelor's degree in Electronics, Communication, and Information Engineering at Thapathali Campus. He is currently in the final year of his degree. His interests lie in the fields of Data Mining, Computer Vision, and Deep Learning.