Lazy Learner in High Stakes Environments: Assessing k-NN Accuracy and Transparency in Crime, Water, and Cybersecurity Audits

Perlin Precious S. Sasil
College of Computer Science and Engineering
Jose Rizal University
Makati City, Philippines
perlinprecious.sasil@my.jru.edu

Abstract*—This report will evaluate the efficiency of the predictions of the k-Nearest Neighbor algorithm with classifying the data, primarily using the Fisher's Iris dataset as its baseline. With the learning approach, the study will investigate the measurements of the sepal and petal length and width and how they can be used to leverage and distinguish the Setsosa, Virginica, and Versicolor. The results of the experiment will be able to demonstrate how the model is sensitive to the parameter. Lower values on the other hand will have a consistent perfect accuracy score of 1.0. However, having a higher neighbor count will lead to the confusion of the model and the performance dropping to a 0.9 because of the overlap of the species. This will also tackle the k-NN usage when it comes to public sectors. Crime rate predictions, municipal water safety, and cyber security intrusion detection. The nature of k-NN being an instance based nature type allows it to provide a more transparent and audible decision making application for the government.

I. Introduction (*Heading 1*)

In this report, it will explore the use case of machine learning when it comes to predictive modeling when it comes to focusing on data. Machine learning can be referred to as a procedure that designs a piece of data that was inputted to a predetermined category. In this study, the k-Nearest Neighbors algorithm will be implemented to predict the Iris Flower based on the physical measurements (Sepal and Petal). The Iris dataset was introduced by the British biologist and statistician Ronald Aylmer Fisher during his 1063 paper titled "The use of Multiple Measurements in taxonomic problems". This dataset consisted of 150 instances, and was distributed with the provided 50 samples from each species Iris Setosa, Iris Virginica, and Iris Versicolor. The four distinct features became an example for measuring centimeters. They are called sepal length, sepal width, petal length, and petal width.[1]

When it comes to implementing the K-NN algorithm, it utilizes being the primary classifier and has a supervised machine learning algorithm used for both classification and regression. Being known as the "Lazy Learner", it stores the dataset that it had obtained form training and will postpone the important computation until it will set a determined time of prediction. Classified as an instance, It identifies the k nearest neighbors with the same class that is based on a similar measurement. While predictive accuracy has its primary metric when it comes to evaluating the machine learning systems that were used to instigate the empirical data. Having supervised machine learning algorithms with both the regression and classification or having transparency with the classifier that can be equally important in the practical applications. The contents of this report will evaluate the gathered performance of the k-NN model when it comes to using metrics that have an overall accuracy and confusing matrix with identifying specific patterns of misclassification with the species. [2]

II. Methodology

A. Importing and Data Acquisition

WHY: In this part, from what we have obtained from sklearn (Scikit-learn) for reference. With the Iris dataset, it can be used because of its benchmark for the taxonomic classification to identify iris species based on morphological measurements. It was introduced by Ronald Fisher in 1936, and it had proved that the species would be distinguished using the multiple measurement systems.

HOW: With the k-NN model in this code, the process begins with having the data acquisition and the selection feature. Raw measurements will be separated into 2, the X (feature) and the y (targets). It will then have a 80/20 train-test split to have an ensured consistency with the measurements and can prevent overfitting. Being a no-limit "lazy learner", It will calculate the Euclidean distance that will categorize unseen data that will be based on five closest neighbors to obtain a higher accuracy of 96%.

```
from sklearn.datasets import load_iris

data = load_iris()
```

B. Preprocessing & Feature Selection

WHY: The species are defined by four variables, the sepal length and width, and the petal length and width. The specimens will coordinate in a four dimensional space. Due to us being unable to visualize the four-dimensional space, we can start to break apart the numerical features to their labeled categories. The algorithm will be able to calculate the certain distances between each neighbor or points.

HOW: Using certain specific syntax that can possibly assign the four-dimensional measurements and labels for the categories can achieve the separation of the raw data that is turned into a more manageable variable. The line of code x = data.data will extract the feature (x) matrix, and will be able to capture the physical measurements which are sepal

length, sepal width, petal length, and petal width and will be able to organize them into a more two-dimensional array. With y=data.target, it extracts the labels of the species, and will be encoded to be integers and will serve as the needed dependent variable of the model. The separation will be able to calculate the distances of the feature space that the human eye cannot visualize.

```
X = data.data
y = data.target
```

C. Data Splitting

WHY: By using the same data that was used for training that can lead to overfitting, and the model will be able to memorize the noise. By using the 20% split for testing, the model of the "undetermined" flowers. With the use of random_state=0, we can ensure the reproducibility and ensure the shuffle will be the same each time for the verification.

HOW: with the division of the dataset into parts to distinct the training and validation that was used to perform the specific syntax and ensure the model can accurately evaluate the unseen data. The X_Test and X_Train are the sets that we use to create the knowledge of the model. Meanwhile, y_Train and y_Test are the sets that will test the model's accuracy.

The test_size=0.2 part of the code will be the one to dictate the 20% of the data to be used in testing. While the remaining 80% of the code will be used to train the model. random_state = 0 on the other hand will be the one to do the random generator to make sure that we will get the split properly when we run the code. In splitting the data this way, it will prevent it from memorizing the data too well.

```
from sklearn.model_selection import train_test_split
X_Train, X_Test, y_Train, y_Test = train_test_split(X, y, test_size=0.2, random_state = 0)
```

D. Model Initialization

WHY: kNN is a "Lazy Learner" that stores the four-dimensional cords rather than having to find the global formula. The n_neighbors=5 will provide the majority voting system, in this way the model will have a prevention system that will stop them from being tricked by the single outliers. Example of this i s the narrow-sepal Setosa specimen that was mentioned in research of Fisher.

HOW: The initialization and setup of the classification model are executed through specific syntax that prepares the environment for performance evaluation and executes the core algorithm. By using the command from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, the system imports essential tools used to measure model success by comparing predicted species against actual results. The accuracy_score will be the syntax that will calculate the percentage of the correct guesses. It will be the one to show the performance summary of the model. With the classification_report on the other hand will reveal the way the model will identify the individual variants. Lastly, the confusion_matrix will tell which flowers had gotten identified and got mistakenly put into different variants.

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matri
predictions = knn.predict(X_Test)
```

E. Model Training

WHY: By using the knn.predict(x_test), we can determine the linear combinations of the physical measurements that were identified by Fisher and are sufficient for the accurate identification. In this step, measuring the generalization is crucial. The model's way to identify the species correctly was not seen before. The evaluation metrics during this stage are imported because the needed accuracy alone can be very deceptive. The confusion matrix, and the classification report will be needed to identify the specific areas where the model will be seen struggling. Examples of this is the overlapping of the Virginica and Versicolor species.

HOW: The predictions in this generation, the preparation for the model evaluation can be achieved by having a single paragraph logic. Sklearn.metrics import accuracy_score, classification_report, confusion_matrix are necessary to obtain the necessary tools so that the model's success rate can be verified by comparing the species that was being predicted and is to be used against the actual known results. With this, predictions-knn.predict(x_text) will be able to trigger the kNN function process. The algorithm will be able to calculate the distance between the data test specimen that was unseen and will store the training data that can assign the specific label of the species based on the common neighbor.

```
knn.predict(X_Test)
```

F. Prediction and Evaluation

WHY: In this step, the "linear combinations" of the measurements will be enough for the accuracy. Predicting the amount of tests for the set, we will be able to generate that confusion matrix to be able to see the species that are confused. It will usually overlap with Versicolor and Virginica.

HOW: With the quantitative assessment of the model, it will be executed with the concise sequence for evaluating the commands. The sklearn.metrics import accuracy_score, classification_report, confusion_matrix will be the ones pulling the needed statistical functions from the environment to audit the performance of the model. predictions = knn.predict(X_Test) will be needed to apply the kNN logic to try and test the set. Meanwhile, the accuracy=accuracy_score(y_test, predictions) will compare the species that were being predicted against the labels that were true to calculate the given final success ratio. This allows us to move outside of the raw code and into the formal analysis of the models precision and reliability.

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
predictions = knn.predict(X_Test)

accuracy = accuracy_score(y_Test, predictions)
```

III. Results and Experimental Analysis

For the experimentation, the K-nearest Neighbors had achieved a perfect accuracy score of 1.0 due to using the n_neighbors=6, testing all 30 test specimens. With the confusion matrix, it confirmed the performance having shown 11 Setosa, 13 Versicolor and 6 Virginica instances. They were identified without having to misclassify any of them or have any species overlap.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=6)
```

```
Accuracy:  1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         6

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Confusion Matrix:
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

Using K=11 for example also achieved the accuracy of a perfect 1.0 score. It also identified the 30 specimens in the set. Likewise, the confusion matrix also confirms 11 Setosa, 13 Versicolor, and 6 Virginica. They had no errors in classifying the species. This result demonstrates that even with a larger voting pool of 11 neighbors, the species remained perfectly distinct, with no instances of a species like Virginica being misclassified as Versicolor.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=11)
```

```
Accuracy:  1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         6

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Confusion Matrix:
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

However, using the k=67 neighbors, the accuracy had dropped to a significant 0.9. The high neighbor count had caused the model to lose its precision because of the over-generalization. The confusion matrix had revealed the 11 Setosa that was identified correctly, but it struggled with the overlapping features. It misclassified 2 of the Versicolor samples as Virginica, and the 1 Virginica sample as a Versicolor. This had shown that the k=67 is too large for the dataset and it had lead to significant confusion among the species.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=67)
```

```
Accuracy:  0.9
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       0.92      0.85      0.88        13
           2       0.71      0.83      0.77         6

    accuracy                           0.90        30
   macro avg       0.88      0.89      0.88        30
weighted avg       0.91      0.90      0.90        30

Confusion Matrix:
[[11  0  0]
 [ 0 11  2]
 [ 0  1  5]]
```

Using a neighbor count of k=23, the model achieved a perfect accuracy score of 1.0, correctly identifying all 30 specimens in the test set. The confusion matrix confirms this flawless performance, showing that 11 Setosa, 13 Versicolor, and 6 Virginica instances were classified with zero errors. This result demonstrates that k=23 provides a highly stable and effective decision boundary for this dataset, successfully distinguishing between species without the misidentifications seen in higher k-values like 67.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=23)
```

```
Accuracy:  1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         6

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Confusion Matrix:
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

The model initialized with k=9 neighbors achieved a perfect accuracy score of 1.0, successfully identifying all 30 specimens in the test set without a single error. The confusion matrix confirms this flawless performance, showing that 11 Setosa, 13 Versicolor, and 6 Virginica instances were correctly classified with zero instances of species overlapping.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=9)
```

```
Accuracy:  1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         6

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Confusion Matrix:
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

IV. REAL-WORLD APPLICATIONS

Crime Rate Prediction; As detailed in the IRJET research paper by Umamaheswaran et al., law enforcement agencies use kNN to predict crime hotspots and types. By analyzing variables like time, date, and geographic coordinates (X and Y longitude/latitude), the model identifies "neighboring" historical incidents to forecast likely future criminal activity, allowing for faster response times and better resource allocation. [4]

Municipal Water Safety Audits: The kNN model achieved a superior accuracy of 95% in predicting the Water Quality Index, outperforming the Decision Tree by providing a more reliable classification of safety levels for educational institutions. By analyzing six critical physico-chemical factors, the algorithm identifies "neighbors" in historical data to accurately categorize water status from "High Quality" to "Unsuitable to use." This model provides the high level of algorithmic transparency required for government safety audits, as its predictions are based on traceable chemical similarities rather than "black box" logic. [5]

Intrusion Detection: The kNN classifier effectively identified 100% of intrusion attempts in the training data by treating system calls as "words" and comparing their frequencies to established normal program behavior. Using an optimal value of k=10, the model achieved a high detection rate with a minimal false positive rate of 0.44%, demonstrating superior precision in distinguishing malicious activity from routine system operations. This approach offers significant algorithmic transparency for government cybersecurity, as security analysts can audit the specific historical "neighbors" that triggered an alert rather than relying on an opaque decision-making process.[6]

V. CONCLUSION

kNN algorithm can demonstrate exceptional effectiveness on using the Iris Dataset. It can achieve a perfect prediction of 1.0 and can optimize each k-values with identifying the species by using the physical measurements. However, when it comes to higher k-values, it will start to struggle due to its attempts to copy the original normal data. Example of this is the process table intrusion attacks. Despite having limitations, the algorithm remains its superiority when it comes to government use because of its transparency and can allow the auditing and have clearer explanations with clear and instant base logic.

REFERENCES

[1] S. Patrick, “Classification of Iris Flower Dataset using Different Algorithms,” *Int. J. Sci. Res. in Mathematical and Statistical Sciences*, vol. 9, no. 6, pp. 1–6, Dec. 2022.

[2] C. Lakshmi Devasena, “Effectiveness Evaluation of Rule Based Classifiers for the Classification of Iris Data Set,” *International Journal of Computer Science & Information Technology (IJCSIT)*, vol. 6, no. 3, pp. 253–268, June 2014.

[3] J. Sebastien, “Visualizing datasets,” *The Data Frog*, 2020. [Online]. Available: https://thedatafrog.com/en/articles/visualizing-datasets/

[4] V. Pednekar, S. Mahale, and G. Gadhave, “Crime Rate Prediction using KNN,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 5, pp. 3120–3123, May 2020.

[5] V. Queen Jemila, M. Dhanalakshmi, and M. Amutha, “Water Quality Prediction Using Decision Tree and KNN,” *International Journal of Innovative Science and Research Technology (IJISRT)*, vol. 9, no. 1, pp. 137–141, Jan. 2024.

[6] Y. Liao and V. R. Vemuri, “Use of K-Nearest Neighbor classifier for intrusion detection,” *Computers & Security*, vol. 21, no. 5, pp. 439–448, 2002.