Summary Report of Findings of Online Cheating Classification Using K-Nearest Neighbors (KNN)

Refino Kashi Kyle Estudillo & Kirsten Kiara Tabasa

December 20, 2024

Video Presentation Link: `Google Drive Link Here`

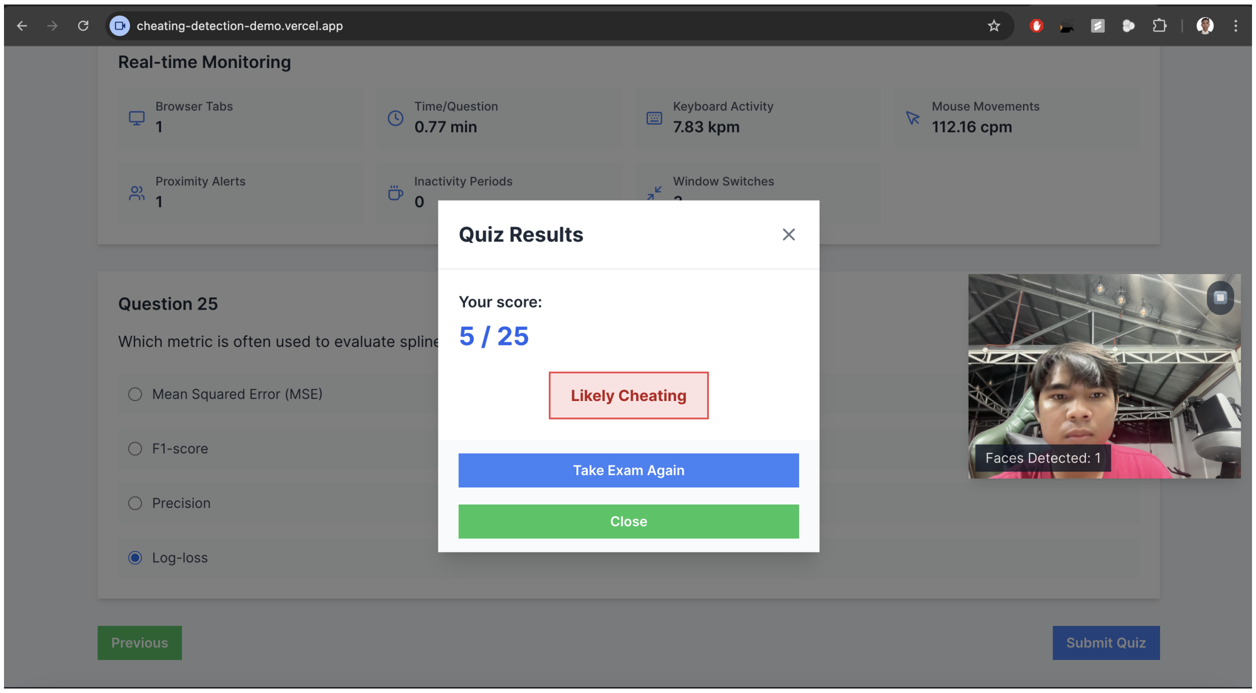
I. Context and Objective

This project demonstrates a custom implementation of the k-Nearest Neighbors (KNN) algorithm to detect potential cheating behaviors in an online examination setting. The classification task aims to distinguish between:

1. Class 1 (Cheating Detected): Strong evidence of cheating behavior.

2. Class 0 (No Cheating Detected): Absence of significant cheating indicators.

A corresponding web application, available at **https://cheating-detection-demo.vercel.app/**, allows real-time monitoring and classification of student behavior, integrating these findings into a user-friendly quiz interface.



Real-time Monitoring

Browser Tabs
1

Time/Question
0.77 min

Keyboard Activity
7.83 kpm

Mouse Movements
112.16 cpm

Proximity Alerts
1

Inactivity Periods
0

Window Switches
2

Question 25

Which metric is often used to evaluate spline

- Mean Squared Error (MSE)
- F1-score
- Precision
- [✓ Log-loss]

Quiz Results

Your score:

5 / 25

Likely Cheating

Figure 1: Illustration of the cheating detection system interface

II. Data and Features

The dataset includes several behavioral and contextual features such as:

- **Number of Browser Tabs Open**: Total number of browser tabs.
- **Time Per Question**: Average time spent per question.
- **Keyboard Activity**: Keypress frequency.
- **Mouse Movements/Clicks**: Frequency of mouse interactions.
- **Proximity Alerts**: Counts of other faces detected in webcam feed.
- **Inactivity Periods**: Duration of no detectable user activity.
- **Window Focus/Switching**: Count of navigations away from the exam window.

All features were normalized to [0,1] to ensure uniform scaling.

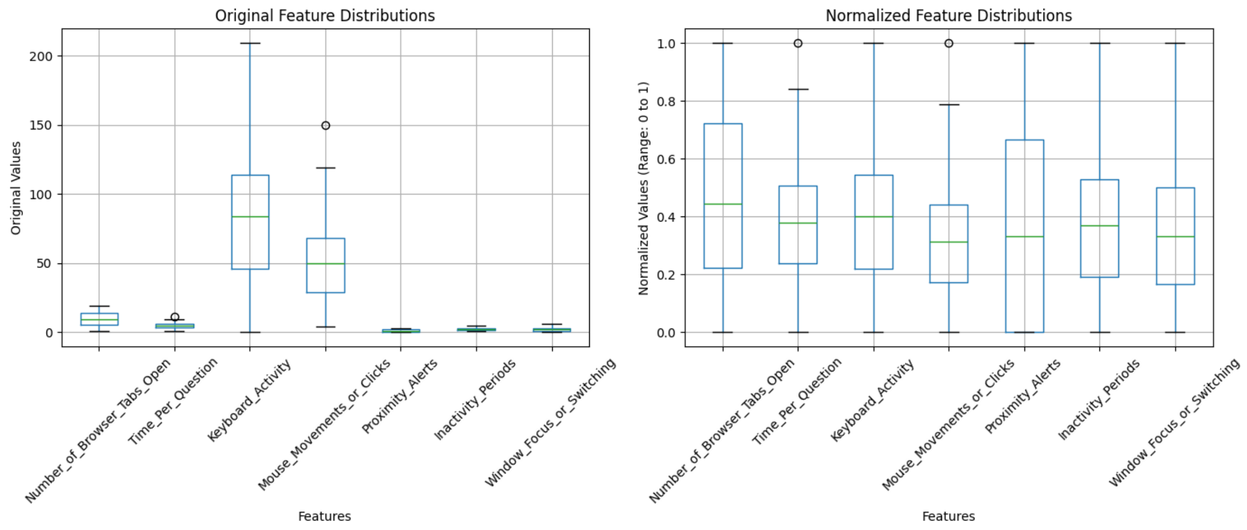


Figure 2: Distribution of normalized features.

III. Methodology

Preprocessing

Data were normalized, and an 80%-20% train-test split was applied to ensure balanced and fair model evaluation.

KNN Implementation From Scratch

The KNN algorithm was implemented directly using `NumPy` without relying on machine learning libraries. The approach involved:

1. **Distance Computation**: For each test instance, we computed the distance to all training instances. Three distance metrics were considered:
   - Manhattan Distance (L_1 norm)
   - Euclidean Distance (L_2 norm)
   - Supremum Distance (L_∞ norm)
2. **Neighbor Selection**: After calculating all distances, the k smallest distances were identified. The corresponding training points formed the neighborhood for the test instance.
3. **Majority Voting**: Within the selected k nearest neighbors, the class that occurred most frequently was assigned to the test instance. This majority vote determines the final predicted class.
4. **Parameter Setting**: We chose $k = 5$ as a balance between noise reduction and preserving local structure. However, k can be tuned based on validation metrics.

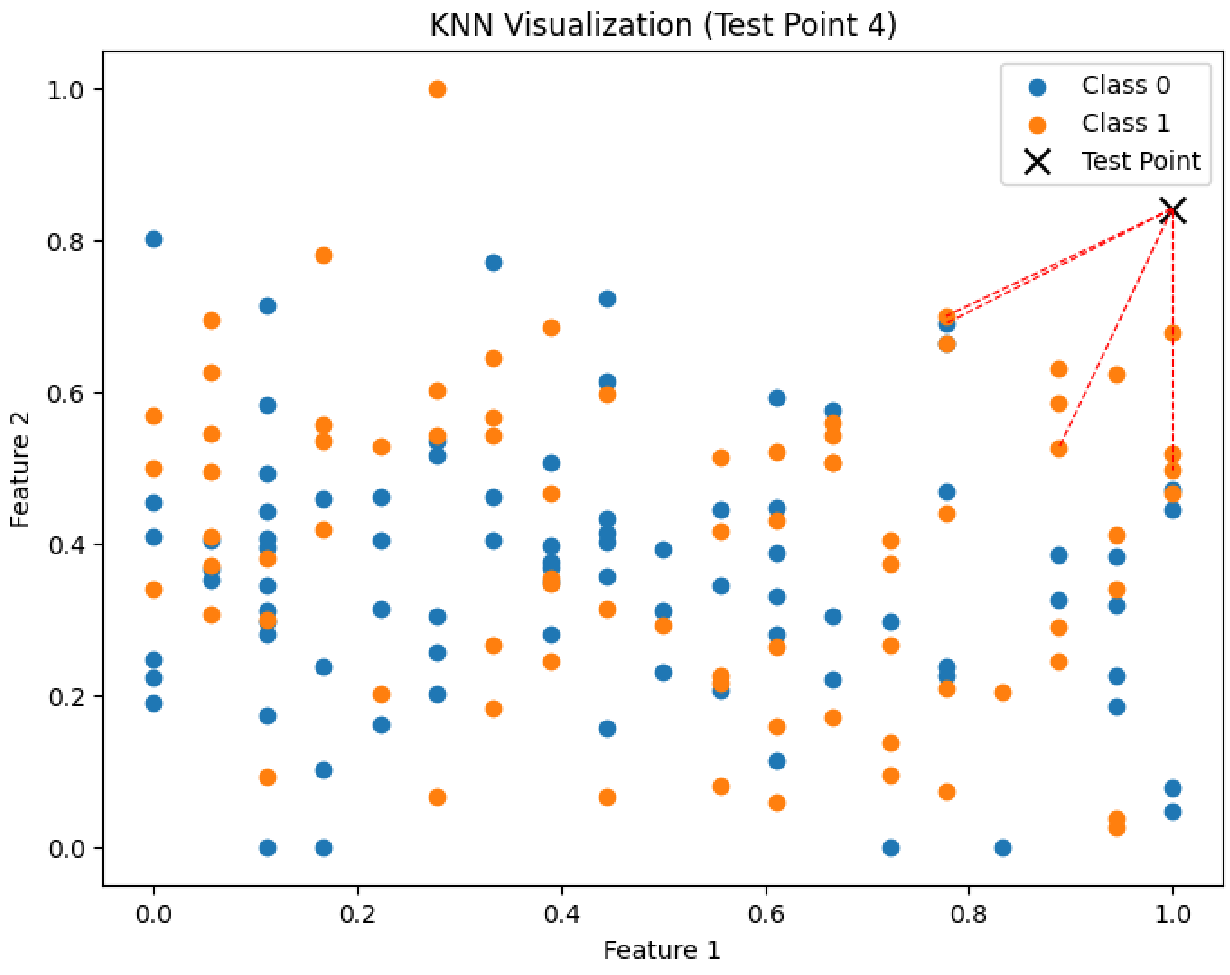


Figure 3: Illustration of KNN decision boundaries in a simplified 2D feature space.

Model Evaluation

Performance was assessed using:

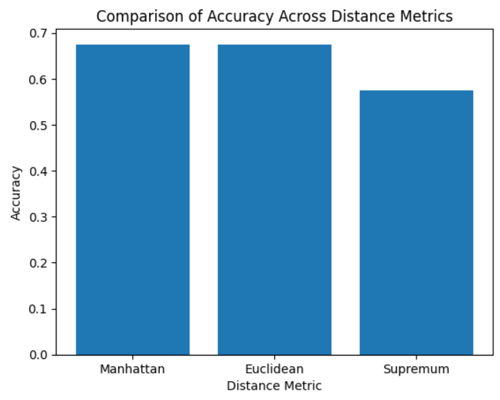
- **Accuracy**: Overall correct classifications.
- **Precision**: Correctness of positive (cheating) predictions.
- **Recall**: Ability to identify all actual cheating cases.
- **F1-Score**: Harmonic mean of precision and recall.

IV. Results

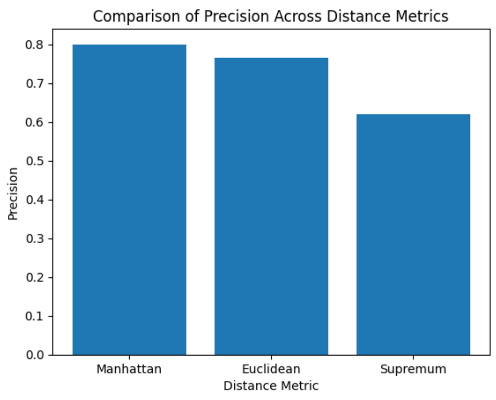
Key performance metrics for $k = 5$ neighbors are summarized in Table 1.

Distance Metric	Accuracy	Precision	Recall	F1-Score
Manhattan	$\approx 67.5\%$	0.80	0.55	0.65
Euclidean	$\approx 67.5\%$	0.76	0.59	0.67
Supremum	$\approx 57.5\%$	0.62	0.59	0.60

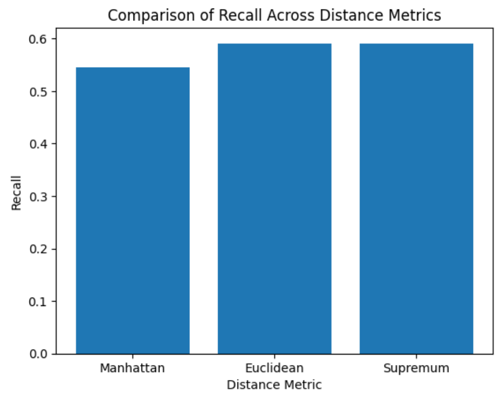
Table 1: Key performance metrics for $k = 5$ neighbors across different distance metrics.



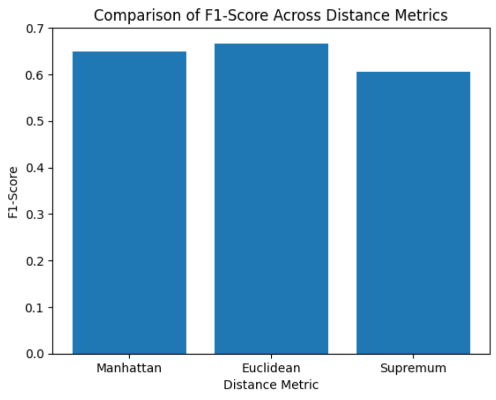
(a) Accuracy



(b) Precision



(c) Recall



(d) F1-Score

Figure 4: Visualization of each metric (Accuracy, Precision, Recall, and F1-Score) across different distance metrics.

V. Analysis and Interpretation

The Manhattan distance metric, which computes the sum of the absolute differences between feature values, is particularly sensitive to variations in features with larger ranges, such as Keyboard Activity and Mouse Movements or Clicks. This sensitivity allowed the model to achieve the highest precision (0.80), indicating its effectiveness in minimizing false positives and correctly identifying cheating instances. However, its lower recall (0.55) suggests that it missed a significant number of true cheating cases. Consequently, the F1-score of 0.65 reflects a moderate trade-off between precision and recall. While Manhattan distance is effective in detecting true positives, it struggles to capture a more balanced performance in identifying all instances of cheating.

Euclidean distance, which squares the differences between feature values, emphasizes larger differences more than Manhattan distance. This metric achieved the same overall accuracy as Manhattan (0.675), with slightly lower precision (0.76) but higher recall (0.59). The higher recall indicates that Euclidean distance is better at capturing true positives, reducing the number of missed cheating cases. The resulting F1-score (0.67) reflects an improved balance between precision and recall compared to Manhattan distance. This makes Euclidean distance a more suitable choice when both false positives and false negatives are of concern. Its ability to balance cumulative differences across features contributes to its robust performance.

Supremum distance, which focuses exclusively on the maximum difference between feature values, disregards cumulative contributions from other features. This limitation resulted in the poorest overall performance among the three metrics. With an accuracy of (0.575), it underperformed in classifying test points correctly. Its precision (0.62) and recall (0.59) are both lower than those of Manhattan and Euclidean metrics, indicating that it neither effectively reduces false positives nor captures true positives. The F1-score of 0.60 further highlights its inability to balance precision and recall adequately. Supremum distance may overlook critical variations in smaller features, making it less effective in datasets where nuanced feature contributions play a significant role.

VI. Practical Implementation

A full-stack web application was developed:

GitHub Repository: `https://github.com/rgestudillo/cheating-detection-demo`

- **Frontend**: Next.js, TypeScript, and Tailwind CSS.
- **Backend**: FastAPI and Python.
- **Real-Time Monitoring**: Browser APIs track tab switching, inactivity, and webcam proximity alerts.
- **Feedback**: Immediate classification results integrated into the user interface.

VII. Conclusion

The evaluation of the KNN model with different distance metrics reveals how dataset characteristics and distance metric properties influence classification performance. The dataset's feature scales significantly impacted the results, as features like Keyboard Activity and Mouse Movements or Clicks have large numerical ranges, dominating distance calculations. Metrics sensitive to these scales, such as Manhattan and Euclidean distances, demonstrated better performance. Additionally, a slight class imbalance in Cheating Detected cases affected recall, as the smaller number of positive cases made it challenging for the model to capture all instances of cheating. The analysis also highlighted the importance of certain features, with Mouse Movements or Clicks and Keyboard Activity showing strong correlations with cheating behavior, suggesting that distance metrics accounting for these features yielded better outcomes.

Among the three metrics evaluated, Manhattan distance proved most effective for precision-focused tasks where minimizing false positives is critical. Its high precision ensures accurate detection of true positives, although its lower recall indicates some missed cheating cases. Euclidean distance, on the other hand, provided a better balance between precision and recall, making it more suitable for general applications where both false positives and false negatives must be minimized. Supremum distance, however, performed the poorest due to its reliance on the maximum feature difference, which neglects cumulative variations critical for accurate classification. This limitation makes it unsuitable for datasets where subtle variations across multiple features significantly influence classification.

In conclusion, the choice of distance metric should align with the specific goals of the application. Manhattan distance is ideal for tasks prioritizing precision, Euclidean distance offers a versatile balance for broader use cases, and Supremum distance is less effective for datasets with complex feature interactions. Future improvements, such as addressing class imbalance, refining feature scaling, and incorporating additional or transformed features, could further enhance the model's performance across all metrics. These insights emphasize the importance of tailoring distance metric selection to the dataset's characteristics and the application's requirements.