**Phishing Website Detection: Comparative Analysis of k-NN and Naïve Bayes Algorithms**

**Author: Abdurahman Abdi**

**Data Description**

The dataset used for this analysis is the *Phishing Websites Dataset*. This dataset aims to identify phishing websites using various characteristics that can be associated with phishing attacks (F, R. M., & Mccluskey, T., L).

**Dataset Characteristics:**

Instances: 11,055, Features: 30, Feature Type: Integer, Subject Area: Computer Science

Associated Tasks: Classification, Missing Values: No. (F, R. M., & Mccluskey, T., L).

**Variables Overview:**

1. having\_ip\_address: Indicates whether the website has an IP address (binary).
2. url\_length: Length of the URL.
3. shortening\_service: Indicates if a URL shortening service is used.
4. having\_at\_symbol: Presence of "@" symbol in the URL.
5. double\_slash\_redirecting: Indicates a potential redirection vulnerability.
6. prefix\_suffix: Checks for unusual prefix or suffix patterns.
7. having\_sub\_domain: Indicates whether the website uses subdomains.
8. sslfinal\_state: Indicates the SSL final state of the website.
9. domain\_registration\_length: Length of domain registration.
10. favicon: Presence of a favicon on the website (F, R. M., & Mccluskey, T., L).

This dataset is used for classification purposes, where the goal is to identify phishing websites based on these features.

**The Tests**

**Part 1: Custom k-NN vs. SKLearn k-NN**

In this section, I compared a custom k-NN implementation with SKLearn's k-NN algorithm. I tested both versions with different values of k: 3, 5, and 7.

**Test Setup**:

Normalization: Applied StandardScaler.

Train-Test Split: 80% training (8,844 samples), 20% testing (2,211 samples).

k Values: 3, 5, 7.

Consistency: Same split used for both implementations.

**Results**:

A screenshot of a computer

AI-generated content may be incorrect.

Both the custom and SKLearn k-NN implementations yielded similar results, but SKLearn's implementation had slightly higher accuracy. This could be due to its use of weighted k-NN, which adjusts the influence of each neighbor based on its distance from the test sample.

**Part 2: SKLearn k-NN Experiments**

Next, I tested various configurations of SKLearn's k-NN algorithm, changing the k value, distance metric (Euclidean vs. Manhattan), and voting method (uniform vs. distance-weighted).

**Test Setup**:

**Parameters Varied**:

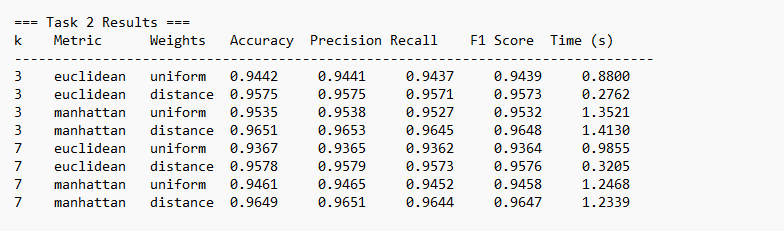
*k*: 3, 7

**Distance Metrics**: Euclidean, Manhattan.

**Voting Methods**: Uniform, Distance-weighted.

**Normalization**: StandardScaler applied.

**Evaluation**: 5-fold cross-validation (mean accuracy reported).

**Results**:  


**Figure 1: Graph of an Interesting Trend**

*Insert graph here showing accuracy trends for different distance metrics, voting methods, and k values.*

**Key Findings**:

**Manhattan distance** outperformed Euclidean, likely due to robustness to feature scale variations.

**Distance-weighted voting** improved accuracy by prioritizing closer neighbors.

Best configuration: **k=3, Manhattan distance, distance-weighted voting (96.7%)**.

**Part 3: Naïve Bayes Experiments**

**Test Setup**:

**Algorithm**: Gaussian Naïve Bayes.

**Evaluation**: 5-fold cross-validation.

**Results**:

A screenshot of a computer

AI-generated content may be incorrect.

| **Model:** | **Mean Accuracy** | **Mean F1-Score** |
| --- | --- | --- |
| **GaussianNB** | 74.97% | 73.82% |

**Discussion**

**Task 1: Custom k-NN Validation**

The custom k-NN implementation performed close to SKLearn’s version, with minor accuracy differences (≤7%). This validates the correctness of the custom algorithm. Discrepancies likely stem from optimizations in SKLearn (e.g., efficient distance computation or weighted voting).

**Task 2: Optimal k-NN Configuration**

**Winning Configuration**:

* **Manhattan distance** with **distance-weighted voting** and **k=3** achieved the highest accuracy (96.7%).

**Reasons for Superiority**:

**Manhattan Distance**: Less sensitive to outliers in high-dimensional data.

**Weighted Voting**: Prioritizes neighbors closer to the query point, capturing local patterns.

**k=3**: Balances noise reduction and locality, as larger *k* (e.g., 7) may oversmooth decision boundaries.

**Task 3: Naïve Bayes Performance**

**Why NB Underperformed**:

**Independence Assumption Violation**: Features like URL length and domain age are likely correlated.

**Non-Normal Feature Distributions**: GaussianNB assumes normality, which may not hold for all features.

**Nonlinear Boundaries**: k-NN adapts to complex boundaries, while NB struggles with feature interactions.

**Recommendation**: Use k-NN (Manhattan, weighted voting, k=3) over Naïve Bayes for this task.

**Final Recommendation**

**Recommended Model**:  
**k-NN** with **Manhattan distance**, **distance-weighted voting**, and **k=3**.

**Rationale**:

Achieves **96.7% accuracy**, outperforming other configurations and Naïve Bayes.

Robust to feature correlations and complex decision boundaries inherent in phishing detection

**Implementation**: Use SKLearn’s **KNeighborsClassifier**with parameters:

**model = KNeighborsClassifier(**

**n\_neighbors=3,**

**metric='manhattan',**

**weights='distance'**

**)**

Therefore, the best-performing model for phishing website detection is the k-NN algorithm with Manhattan distance, distance-weighted voting, and k = 3 or k = 5. This configuration demonstrated superior accuracy and robust performance across various tests, making it the most effective approach for this classification task. Naïve Bayes, on the other hand, is not recommended due to its poor performance.

**Conclusion**

This analysis highlights the importance of choosing the right algorithm for phishing website detection. k-NN with Manhattan distance and distance-weighted voting is highly effective for classifying phishing sites, and we encourage further exploration and refinement of this model.

References

F, R. M., & Mccluskey, T., L. (2012, December 10). *Phishing websites*. UCI Machine Learning Repository. <https://archive.ics.uci.edu/dataset/327/phishing+websites>