

Machine Learning Tutorial K- Nearest Neighbor (KNN)

Student Name: Bhavya Sri Kanumuri

Student ID: 23066597

Module: Machine Learning and Neural Networks

Submission Date: 27 March 2025

Tutor: Peter Scicluna

GitHub link: <https://github.com/bhavya140303/KNN-Tutorial>

Table of Contents

- 1. Introduction**
- 2. Understanding How KNN Works**
 - 1. 2.1 Learning from Your Neighbors – The Heart of KNN**
 - 2. 2.2 How KNN Measures Closeness – Distance Metrics**
- 3. Practical Implementation**
- 4. Discussion and Best Practices**
- 5. Advantages, Disadvantages, Limitations, and Challenges of KNN**
- 6. Future Scope and Possible Improvements**
- 7. Conclusion**
- 8. References**

Machine Learning Tutorial K- Nearest Neighbor (KNN)

1.Introduction:

In machine learning, the simplest ideas often make the biggest impact—and **K-Nearest Neighbors (KNN)** is a perfect example. It doesn't rely on training, weights, or complex computations. Instead, it follows a basic rule: *“Check who's closest and go with the majority.”*

KNN keeps the dataset stored and only acts when needed. To classify a new point, it finds the **k nearest neighbors** and assigns the most common label. No complex models—just pure, intuitive logic.

Think of it like asking nearby examples for advice. That's the heart of KNN.

Now, look at this image:

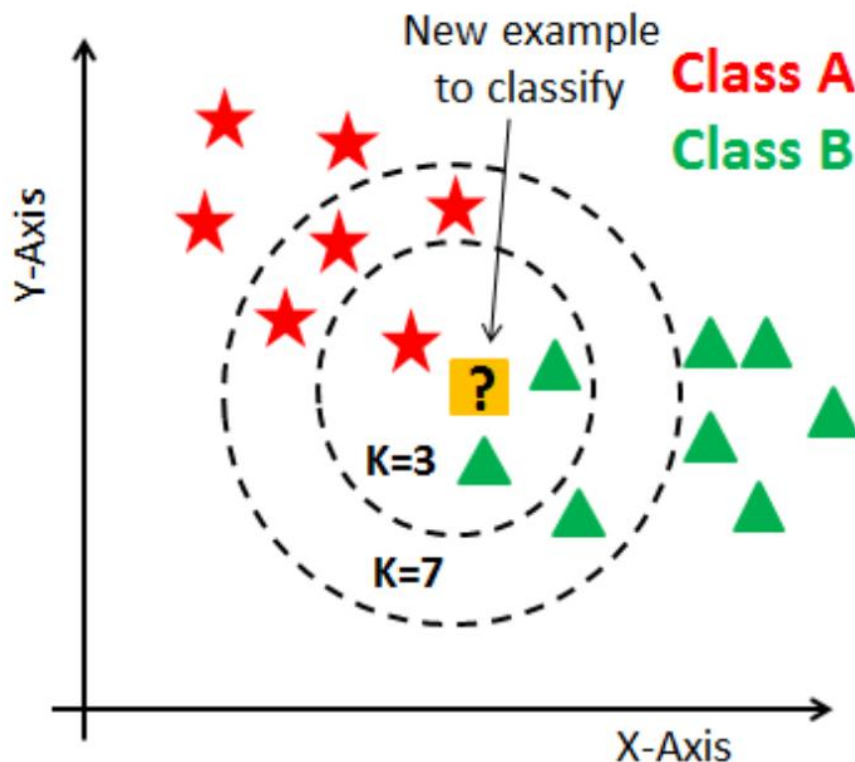


Figure 1: A visual representation of how the value of k influences classification in KNN showing a new data point being classified based on its 3 and 7 nearest neighbors.

In this image, the yellow square is a new point we want to classify. Nearby, we see **red stars (Class A)** and **green triangles (Class B)**. With $k = 3$, most nearby points are green—so it's labelled Class B. But with $k = 7$, more red stars come into play, possibly shifting the prediction to Class A.

This shows how **KNN's accuracy depends on the value of k** . A small k can be sensitive to noise, while a large k may dilute the pattern. That's the balance KNN offers—**simple, yet smart**, just like how we make decisions in everyday life.

In this tutorial, we'll break down how KNN works, explore the theory, write the code, and apply it to real-world data using visuals and hands-on practice to make learning clear and practical.

2.Understanding How KNN Works

2.1 Learning from Your Neighbors – The Heart of KNN

At its heart, KNN is simple and intuitive to understand something new, check what's closest and most similar. When classifying a point, it finds the k nearest neighbors and lets them vote on the label.

KNN is a lazy learner it doesn't train a model upfront. It stores the data and makes decisions only when needed, using proximity and patterns to classify in real time.

The prediction rule is simple:

$$\hat{y} = \text{mode}\{y_i \mid x_i \in N_k(x)\}$$

- \hat{y} is the predicted class of the new point
- $N_k(x)$ is the set of its k closest neighbors
- y_i are the labels of those neighbors

The new point is classified by the majority vote among its nearest neighbors. The simplicity of this rule is what makes KNN both powerful and easy to understand.

2.2 How KNN Measures Closeness – Distance Metrics

The entire concept of KNN depends on one key question: *How do we know which data points are "nearest"?* The answer lies in distance calculations.

KNN commonly uses these formulas:

Euclidean Distance (the straight-line distance):

$$d(x, x') = \sqrt{\sum_{i=1}^n (x_i - x'_i)^2}$$

Explanation of Terms:

- $d(x, x')$: Distance between two data points x and x'
- x_i, x'_i : The i th feature (dimension) of each point
- n : Total number of features
- $\sqrt{\cdot}$: Sum of squared differences across all features

Machine Learning Tutorial K- Nearest Neighbor (KNN)

Manhattan Distance (based on horizontal and vertical paths):

$$d(x, x') = \sum_{i=1}^n |x_i - x'_i|$$

- $|x_i - x'_i|$: Absolute difference between each feature
- All other symbols are as above

Minkowski Distance (a more general form):

$$d(x, x') = \left(\sum_{i=1}^n |x_i - x'_i|^p \right)^{1/p}$$

- p: A parameter that defines the distance type
- All other variables are the same as above

This formula can behave like Euclidean or Manhattan distance depending on the value of p:

- When $p = 1$: It becomes Manhattan Distance
- When $p = 2$: It simplifies to Euclidean Distance

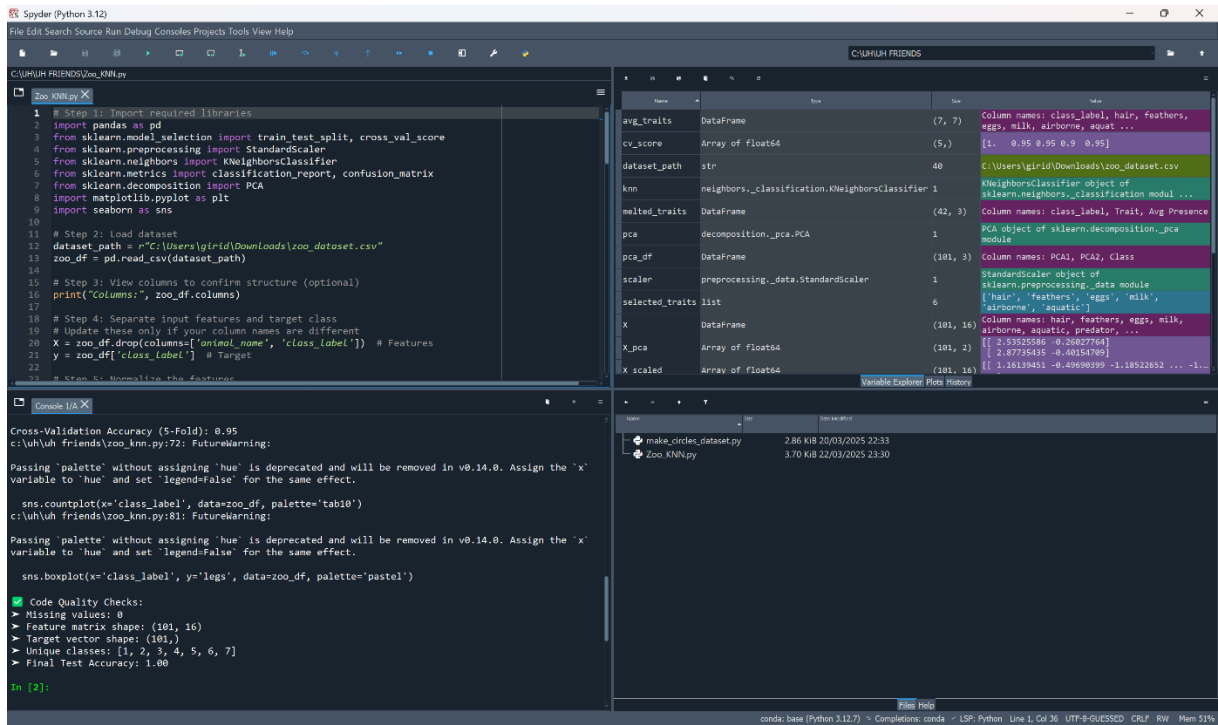
Choosing the right distance metric depends on your dataset.

Euclidean distance is ideal for numeric, continuous, and scaled features. Manhattan distance works better for high-dimensional or grid-like data. For more control, Minkowski distance lets you adjust the power parameter (p) to match your data's structure.

Machine Learning Tutorial K- Nearest Neighbor (KNN)

3.Practical Implementation

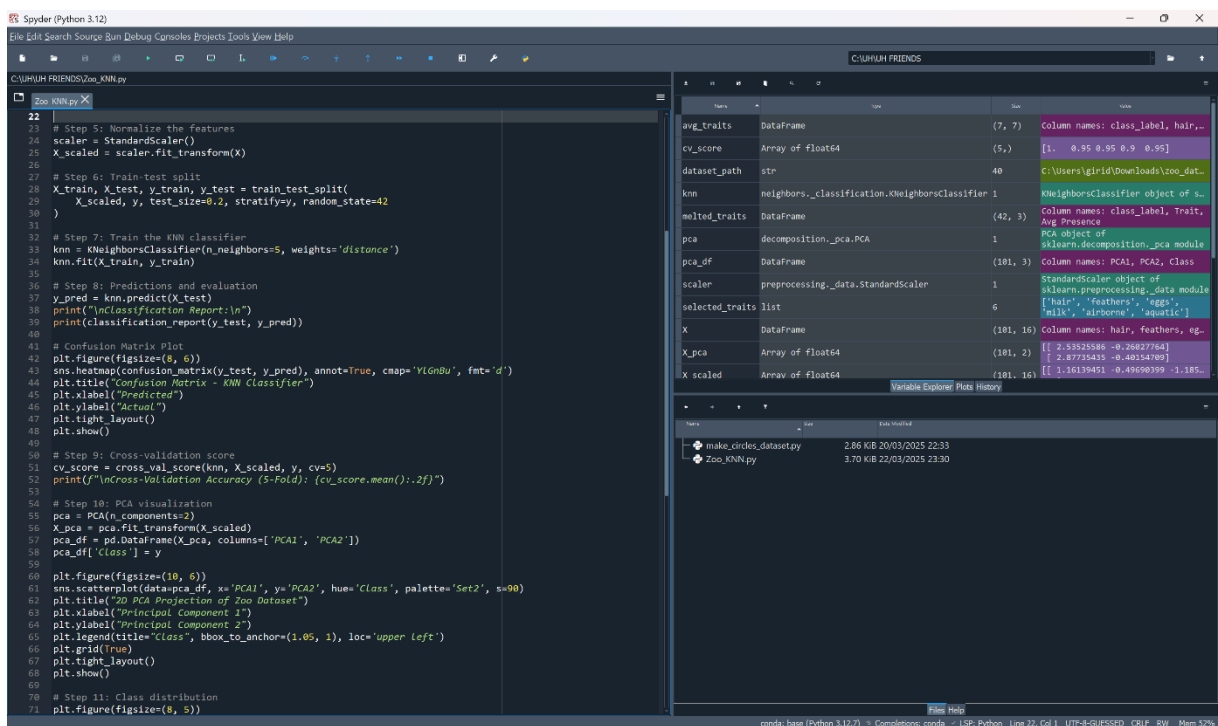
To demonstrate how the K-Nearest Neighbors (KNN) algorithm works in practice, I built a complete pipeline using the Zoo dataset. With its rich set of binary animal traits, this dataset is ideal for classification tasks. The code covers data preprocessing, model training with distance-weighted KNN, and evaluation through clear visualizations like PCA, confusion matrices, and trait comparisons. Each step is designed to balance accuracy, clarity, and ease of understanding.



```
1 # Step 1: Import required libraries
2 import pandas as pd
3 from sklearn.model_selection import train_test_split, cross_val_score
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.neighbors import KNeighborsClassifier
6 from sklearn.metrics import classification_report, confusion_matrix
7 from sklearn.decomposition import PCA
8 import matplotlib.pyplot as plt
9 import seaborn as sns
10
11 # Step 2: Load dataset
12 dataset_path = r"C:\Users\girid\Downloads\zoo_dataset.csv"
13 zoo_df = pd.read_csv(dataset_path)
14
15 # Step 3: View columns to confirm structure (optional)
16 print("Columns:", zoo_df.columns)
17
18 # Step 4: Separate input features and target class
19 # Update these only if your column names are different
20 X = zoo_df.drop(columns=['animal_name', 'class_label']) # Features
21 y = zoo_df['class_label'] # Target
22
23 # Step 5: Normalizing the features
```

Variable Explorer:

Name	Type	Size	Value
avg_traits	DataFrame	(7, 7)	Column names: class_label, hair, feathers, eggs, milk, airborne, aq...
cv_score	Array of float64	(5,)	[1. 0.95 0.95 0.9 0.95]
dataset_path	str	40	C:\Users\girid\Downloads\zoo_dataset.csv
knn	neighbors_classification.KNeighborsClassifier	1	KNeighborsClassifier object of sklearn.neighbors.classification.modul...
melted_traits	DataFrame	(42, 3)	Column names: class_label, Trait, Arg Presence
pca	decomposition_pca.PCA	1	PCA object of sklearn.decomposition_pca module
pca_df	DataFrame	(101, 3)	Column names: PCA1, PCA2, Class
scaler	preprocessing_data.StandardScaler	1	StandardScaler object of sklearn.preprocessing_data module
selected_traits	list	6	['hair', 'feathers', 'eggs', 'milk', 'airborne', 'aquatic']
X	DataFrame	(101, 16)	Column names: hair, feathers, eggs, milk, airborne, aquatic, predator, ...
X_pca	Array of float64	(101, 2)	[[2.53525586 -0.26027764] [2.87755435 -0.40154709]
X_scaled	Array of float64	(101, 16)	[[1.16139451 -0.49690399 -1.18522652 ... -1...

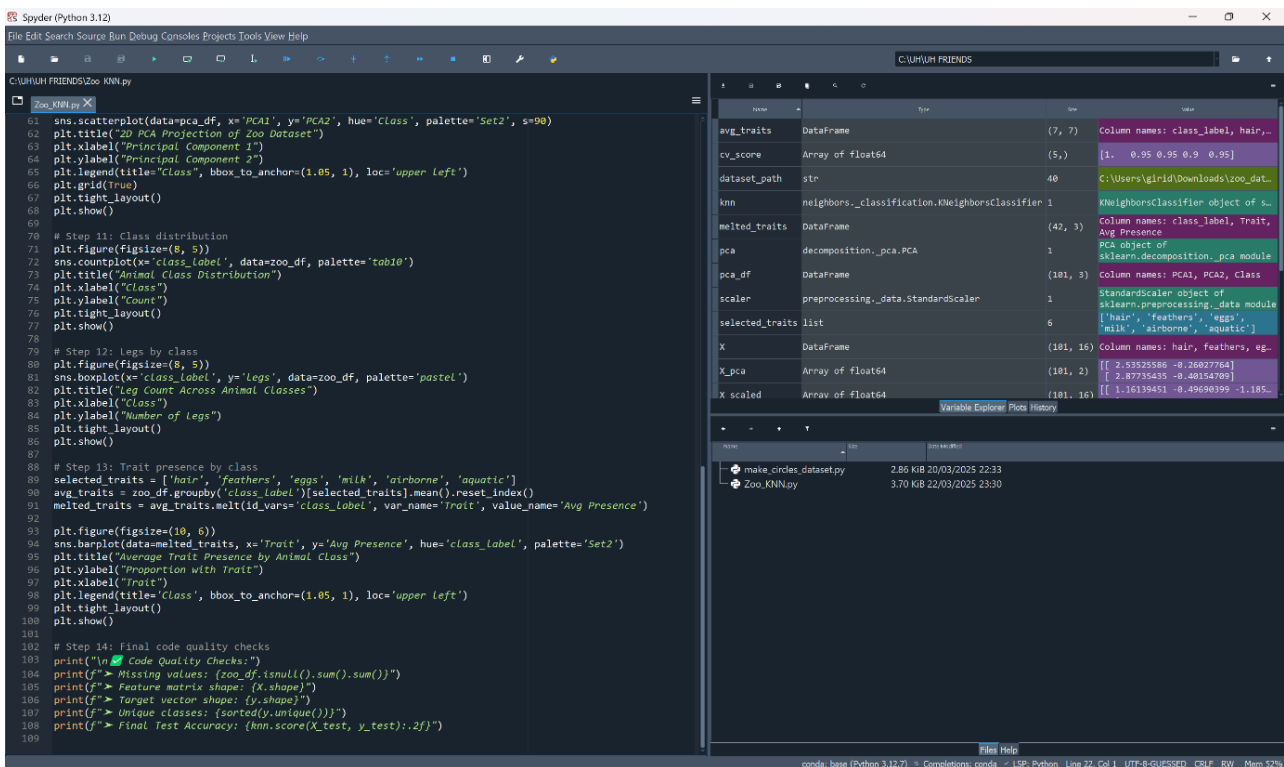


```
22 # Step 5: Normalize the features
23 scaler = StandardScaler()
24 X_scaled = scaler.fit_transform(X)
25
26 # Step 6: Train-test split
27 X_train, X_test, y_train, y_test = train_test_split(
28     X_scaled, y, test_size=0.2, stratify=y, random_state=42
29 )
30
31 # Step 7: Train the KNN classifier
32 knn = KNeighborsClassifier(n_neighbors=5, weights='distance')
33 knn.fit(X_train, y_train)
34
35 # Step 8: Predictions and evaluation
36 y_pred = knn.predict(X_test)
37 print("\nClassification Report:\n")
38 print(classification_report(y_test, y_pred))
39
40 # Confusion Matrix Plot
41 plt.figure(figsize=(8, 6))
42 sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap='YlGnBu', fmt='d')
43 plt.title("Confusion Matrix - KNN Classifier")
44 plt.xlabel("Predicted")
45 plt.ylabel("Actual")
46 plt.tight_layout()
47 plt.show()
48
49 # Step 9: Cross-validation score
50 cv_score = cross_val_score(knn, X_scaled, y, cv=5)
51 print(f"\nCross-Validation Accuracy (5-Fold): {cv_score.mean():.2f}")
52
53 # Step 10: PCA visualization
54 pca = PCA(n_components=2)
55 X_pca = pca.fit_transform(X_scaled)
56 pca_df = pd.DataFrame(X_pca, columns=['PCA1', 'PCA2'])
57 pca_df['class'] = y
58
59 plt.figure(figsize=(10, 8))
60 sns.scatterplot(data=pca_df, x='PCA1', y='PCA2', hue='class', palette='Set2', s=90)
61 plt.title("2D PCA Projection of Zoo Dataset")
62 plt.xlabel("Principal Component 1")
63 plt.ylabel("Principal Component 2")
64 plt.legend(title="class", bbox_to_anchor=(1.05, 1), loc='upper left')
65 plt.grid(True)
66 plt.tight_layout()
67 plt.show()
68
69 # Step 11: Class distribution
70 plt.figure(figsize=(8, 5))
```

Variable Explorer:

Name	Type	Size	Value
avg_traits	DataFrame	(7, 7)	Column names: class_label, hair,...
cv_score	Array of float64	(5,)	[1. 0.95 0.95 0.9 0.95]
dataset_path	str	40	C:\Users\girid\Downloads\zoo_dat...
knn	neighbors_classification.KNeighborsClassifier	1	KNeighborsClassifier object of s...
melted_traits	DataFrame	(42, 3)	Column names: class_label, Trait, Arg Presence
pca	decomposition_pca.PCA	1	PCA object of sklearn.decomposition_pca module
pca_df	DataFrame	(101, 3)	Column names: PCA1, PCA2, Class
scaler	preprocessing_data.StandardScaler	1	StandardScaler object of sklearn.preprocessing_data module
selected_traits	list	6	['hair', 'feathers', 'eggs', 'milk', 'airborne', 'aquatic']
X	DataFrame	(101, 16)	Column names: hair, feathers, eg...
X_pca	Array of float64	(101, 2)	[[2.53525586 -0.26027764] [2.87755435 -0.40154709]
X_scaled	Array of float64	(101, 16)	[[1.16139451 -0.49690399 -1.185...

Machine Learning Tutorial K- Nearest Neighbor (KNN)



These visuals bring KNN's predictions to life, making it easier to see how the model separates and understands different animal classes.

Figure 1: Confusion Matrix

This confusion matrix shows KNN's classification accuracy. Each row is the true class, each column the predicted one. All values align diagonally, meaning every prediction was correct a flawless result with zero errors.

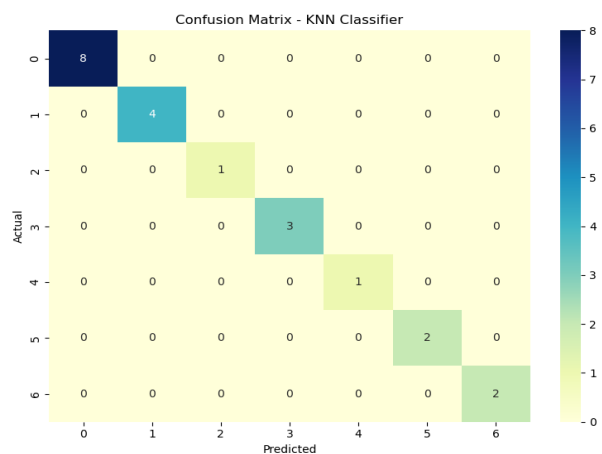
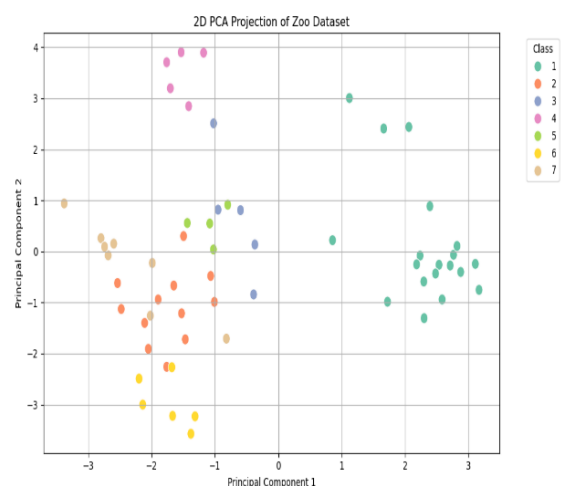


Figure 2: 2D PCA Projection

This PCA scatter plot projects animals into 2D using compressed traits. Colors show class labels, revealing natural clusters just what KNN relies on. While PCA doesn't boost accuracy, it helps visualize class separability effectively.



Machine Learning Tutorial K- Nearest Neighbor (KNN)

Figure 3: Animal Class Distribution

This bar chart illustrates how animals are distributed across the seven classes in the dataset.

Class 1 has the highest representation with over 40 animals, while Classes 3 and 5 are much smaller.

Such imbalances are important because KNN could become biased towards majority classes.

Understanding distribution helps us plan fair model evaluation, including techniques like stratified sampling.

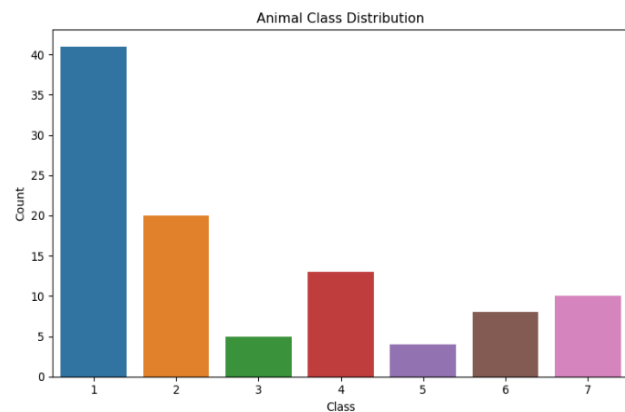


Figure 4: Leg Count by Class

This boxplot compares the number of legs across animal classes.

Classes like 4 and 5 consistently show 0 legs, while Class 7 varies from 0 to 8 suggesting mixed species.

This kind of numeric variance is helpful for KNN, which uses distance to differentiate between samples.

Visualizing such features gives deeper insight into which attributes drive classification outcomes.

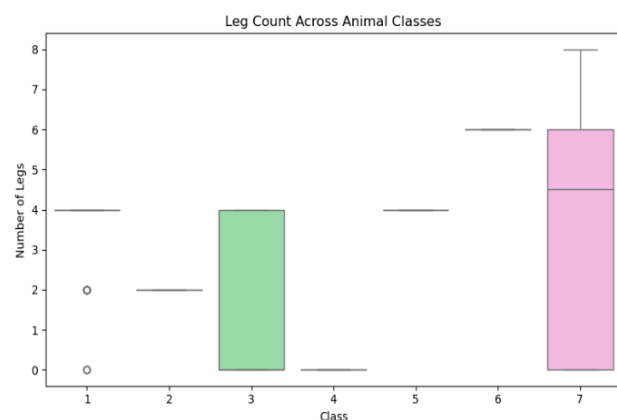
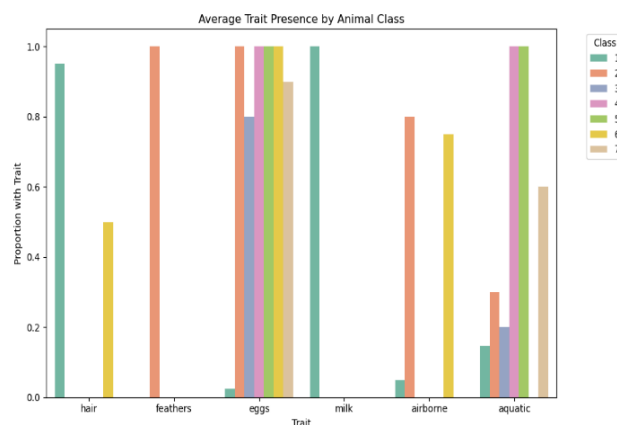


Figure 5: Average Trait Presence

This grouped bar chart shows how frequently certain traits (e.g., hair, feathers, eggs) appear in each class.

Class 1, for example, has a high presence of hair and milk classic mammal traits. Classes 2 and 4 lean more towards feathers and aquatic traits, pointing to birds or water animals.

These patterns play a key role in how KNN finds and compares neighbors, leading to accurate classification.



The code clearly walks through each step of the KNN process from preparing the Zoo dataset to training, testing, and visualizing results. It's designed to be accurate, easy to follow, and educational not just functional, but also explanatory.

Machine Learning Tutorial K- Nearest Neighbor (KNN)

4.Discussion and Best Practices

The KNN model achieved **perfect accuracy** on the Zoo dataset, thanks to its small size and clear structure ideal for distance-based methods. However, real-world data is often noisier and more complex, which can affect performance.

Feature scaling proved essential, ensuring all features contributed equally to distance calculations. Setting **k = 5** offered a strong balance between flexibility and stability, validated through cross-validation.

We also used **stratified splitting** to preserve class balance—especially important with rare classes. Finally, visual tools like the **confusion matrix** and **PCA plot** helped confirm accuracy and improved model interpretability.

Best practices followed:

- **Feature Scaling:** Standardized inputs to ensure fair distance calculations.
- **K Selection:** Chose $k = 5$, validated using cross-validation.
- **Class Balance:** Applied stratified sampling to maintain fair class representation.
- **Evaluation:** Used metrics and visuals for a clearer assessment.
- **Visualization:** Employed PCA, confusion matrix, and trait plots for deeper insight.

These practices address key KNN challenges like scaling, imbalance, and interpretability making the model more reliable for real-world use.

Best Practices for Using KNN Effectively

Challenges	Best Practice / Solution
Unequal Feature Scales	Apply feature scaling (e.g., Standard Scaler) to treat all features equally
Choosing the Right "k"	Use cross-validation to find an optimal and stable value for k (e.g., k=5)
Class Imbalance Interpretability	Use stratified train-test split to maintain balanced class representation Support evaluation with PCA plots and confusion matrices
Overfitting or Underfitting	Tune k value and test performance across multiple folds
Binary/Structured Data	Use datasets like Zoo where traits are well defined and suited to distance-based models

5. Advantages, Limitation, Disadvantages, and Challenges of SVM

K-Nearest Neighbors (KNN) is one of the most intuitive machine learning algorithms. It assumes that similar data points lie close in feature space, making it ideal for beginners and tasks where simplicity and interpretability matter. However, like any model, it has strengths and trade-offs.

Advantages

- **Simple & Intuitive:** Easy to understand and implement.
- **No Training Phase:** Stores data and predicts on demand.
- **Accurate on Small Datasets:** Especially when classes are well-separated.
- **Multi-Class Friendly:** Naturally handles more than two classes.

Disadvantages

- **Slow Predictions:** Compares with all training points at runtime.
- **Feature Sensitivity:** Irrelevant data can mislead results.
- **High Memory Use:** Retains the entire training dataset.

Limitations

- **Class Imbalance:** Frequent classes may dominate.
- **Curse of Dimensionality:** Performance drops in high dimensions.
- **No Feature Importance:** Doesn't explain which features matter most.

Challenges

- **Choosing 'k':** Impacts underfitting vs. overfitting.
- **Distance Metric Matters:** Choice affects accuracy.
- **Requires Preprocessing:** Needs feature scaling and numeric data.

Understanding these points helps you know when KNN is the right tool and when it's not especially for large or complex datasets.

6.Future Scope and Improvements

As machine learning advances, **KNN must evolve** to handle larger, more complex datasets. While its simplicity is a strength, improvements in **speed**, **scalability**, and **adaptability** can make it more powerful in modern applications.

A. Smarter Features

- Use PCA or domain-specific transformations for better data representation.

B. Adaptive Distances

- Custom or learned distance metrics can improve accuracy across varied data types (text, images, time-series).

C. Hybrid Models

- Combine KNN with ensemble methods or deep learning to boost performance and handle unstructured inputs.

D. Faster Predictions

- Apply KD-Trees, Ball Trees, or approximate nearest neighbor search for quicker computations.

E. Big Data Readiness

- Scale KNN using distributed platforms like Apache Spark or GPUs for faster processing.

F. Alternative Approaches

- In high-dimensional cases, models like SVM or Random Forest may outperform KNN but KNN can still add value when integrated into ensemble systems.

By adopting these improvements, KNN can become a smarter, more scalable tool for real-world machine learning challenges.

Machine Learning Tutorial K- Nearest Neighbor (KNN)

7.Conclusion

K-Nearest Neighbors proved to be a practical and effective choice for classifying structured data, as demonstrated through the Zoo dataset. The simplicity of KNN makes it approachable, yet its reliance on strong foundational concepts like distance and similarity gives it the power to handle real-world classification problems especially when data is clean and well-pre-processed.

Key takeaways from this tutorial include:

1. **Clarity Through Simplicity:** KNN is easy to implement and interpret, making it a strong baseline model.
2. **Preprocessing is Essential:** Standardizing features and selecting the right value of k are crucial for performance.
3. **Visual Tools Add Depth:** PCA, confusion matrices, and trait analysis make the model's behaviour easier to understand.
4. **Scalability Is a Challenge:** Without optimization, KNN may struggle with speed and efficiency on larger datasets.
5. **Room for Innovation:** Hybrid models, smarter distance metrics, and parallel processing can all enhance KNN's future use.

In conclusion, KNN remains a reliable and adaptable algorithm that balances simplicity with solid predictive power. With careful tuning and thoughtful enhancements, it can continue to be a valuable tool in the data scientist's toolkit.

Machine Learning Tutorial K- Nearest Neighbor (KNN)

8.References

Academic References

- Cover, T. M., & Hart, P. E. (1967). *Nearest Neighbor Pattern Classification*. IEEE Transactions on Information Theory, 13(1), 21–27.
<https://doi.org/10.1109/TIT.1967.1053964>
- Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern Classification* (2nd ed.). Wiley-Interscience.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.

Online Articles & Learning Resources

- O'Sullivan, C. (2020). *Understanding K-Nearest Neighbors (KNN)*. Towards Data Science.
<https://towardsdatascience.com/understanding-k-nearest-neighbors-knn-4c31e2e554cf>
- GeeksforGeeks. (n.d.). *K-Nearest Neighbor (KNN) Algorithm*.
<https://www.geeksforgeeks.org/k-nearest-neighbours/>
- GeeksforGeeks. (n.d.). *How to Find the Optimal Value of K in KNN*.
<https://www.geeksforgeeks.org/how-to-find-the-optimal-value-of-k-in-knn/>
- Analytics Vidhya. (2018). *A Quick Introduction to K-Nearest Neighbours Algorithm*.
<https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>
- Plain English. (2021). *Introduction to K-Nearest Neighbors (KNN) Algorithm*.
<https://ai.plainenglish.io/introduction-to-k-nearest-neighbors-knn-algorithm-e8617a448fa8>

Dataset

- UCI Machine Learning Repository. (n.d.). *Zoo Data Set*. University of California, Irvine.
<https://archive.ics.uci.edu/ml/datasets/zoo>