K-Nearest Neighbors (KNN) Classifier - Cats & Dogs

Presented by:
Er Zhao, Shenyu Li, Shan Meng, Zifeng Li

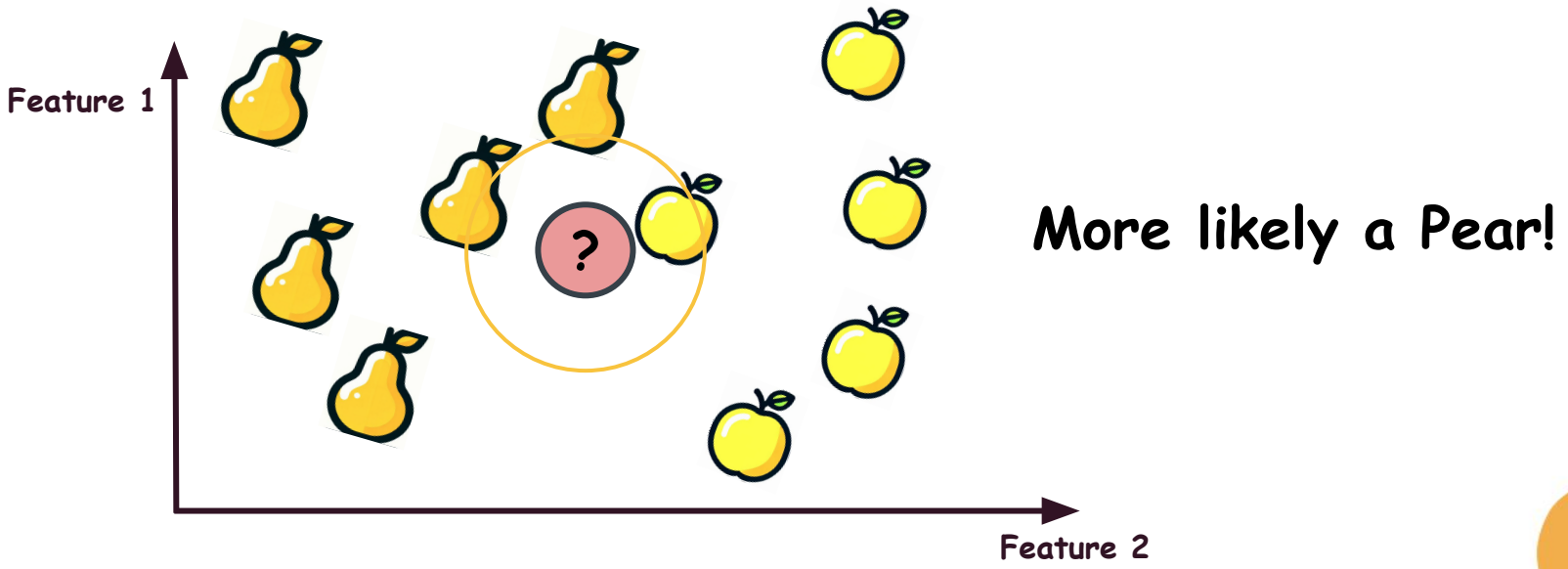
Presentation Outlines

- KNN Introduction
- Code Implementation
- Project Overview

What is K-Nearest Neighbors (KNN)?

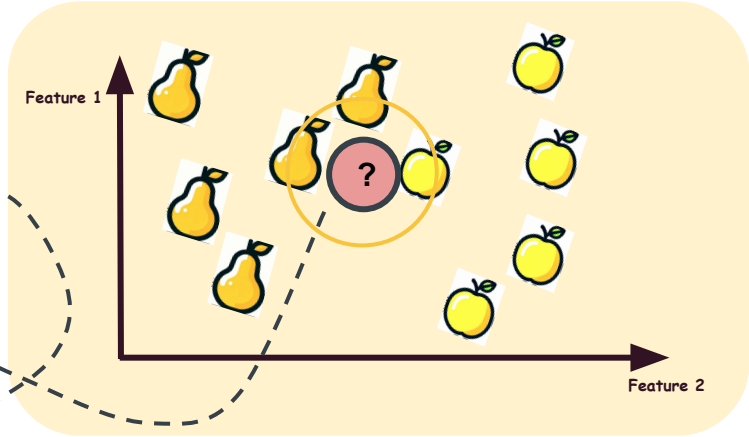
A simple non-parametric, supervised learning classification algorithm.

It classifies data points based on the majority class of their K nearest neighbors.

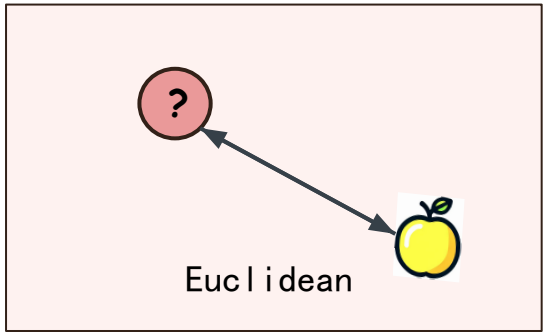


Key Parameters

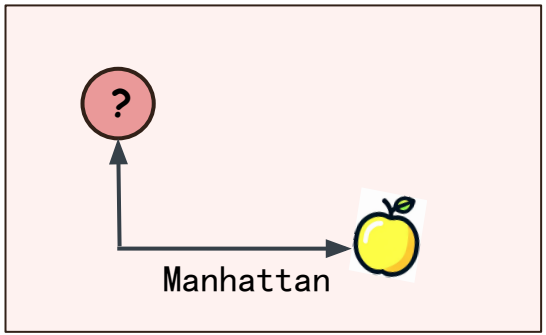
- K → Number of nearest neighbors to consider.



- Distance Metric: Euclidean, Manhattan, or others.



Or



Or Others…

Code Implementation for KNN model

Class KNN

Initialization:
Parameter: k, metrics

Distance Calculation:
Euclidean and Manhattan

Model Training (fit):
Parameter: X_train, y_train

Finding Neighbors(get_neighbours):
Parameter: test_row

Prediction(predict):
Parameter: X_test

Get Score(score):
Parameter: predict, y_test

```python
class KNN:
    def __init__(self, k=3, metric="euclidean"):
        self.k = k
        self.metric = metric

    # Euclidean distance (l2 norm)
    def euclidean(self, v1, v2):
        return np.sqrt(np.sum((v1 - v2) ** 2))

    # Manhattan distance (l1 norm)
    def manhattan(self, v1, v2):
        return np.sum(np.abs(v1 - v2))

    def fit(self, X, y):
        self.X_train = np.array(X)
        self.y_train = np.array(y)

    # Get nearest neighbours and distances
    def get_neighbours(self, test_row):
        distances = []
        for (train_row, train_class) in zip(self.X_train, self.y_train):
            if self.metric == 'euclidean':
                dist = self.euclidean(train_row, test_row)
            elif self.metric == 'manhattan':
                dist = self.manhattan(train_row, test_row)
            else:
                raise NameError("Supported metrics are euclidean and manhattan")
            distances.append((dist, train_row, train_class))
        distances.sort(key=lambda x: x[0])
        return distances[:self.k]

    # Predict using KNN
    def predict(self, X_test):
        preds = []
        for test_row in X_test:
            # Find k nearest neighbours
            neighbours = self.get_neighbours(test_row)

            # Predict the majority class using Counter
            neighbour_classes = [n[2] for n in neighbours]
            majority = Counter(neighbour_classes).most_common(1)[0][0]
            preds.append(majority)
        return np.array(preds)

    # Calculate accuracy score
    def score(self, preds, y_test):
        return 100 * (preds == y_test).mean()
```

Project Introduction

- **Created a machine learning model to differentiate between cats and dogs.**
- **Trained and tested a custom KNN classifier.**

Dataset Introduction

Height

The height of the animal, measured in centimeters (cm).

Weight

The weight of the animal, measured in kilograms (kg).

Length

The length of the animal, measured in centimeters (cm).

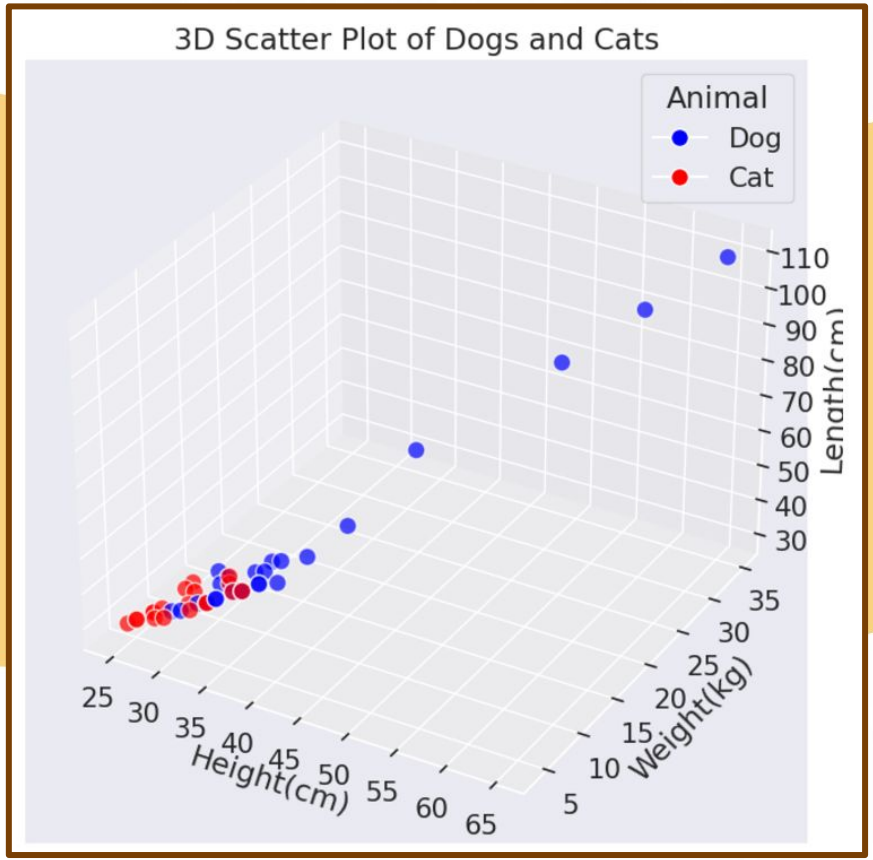
Animal

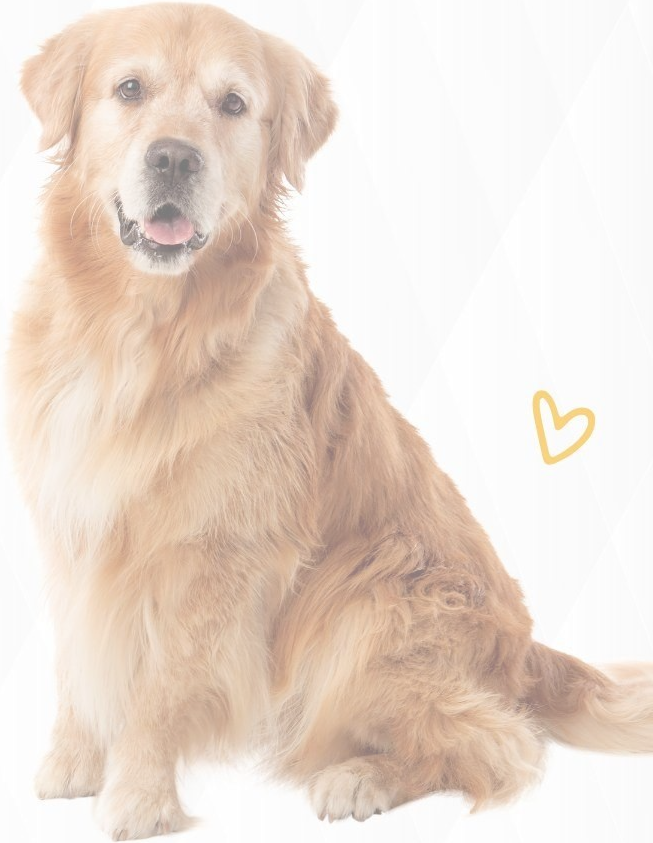
Target Label. The animal type, where 0 represents Cat and 1 represents Dog.

Cats and Dogs Classification Dataset

Data	HEIGHT	WEIGHT	LENGTH	ANIMAL
0	25	4	30	0
1	32	6	38	0
2	38	8	45	1
3	28	5	35	0
4	35	7	42	1

Data Visualization





KNN Model Working Visualization

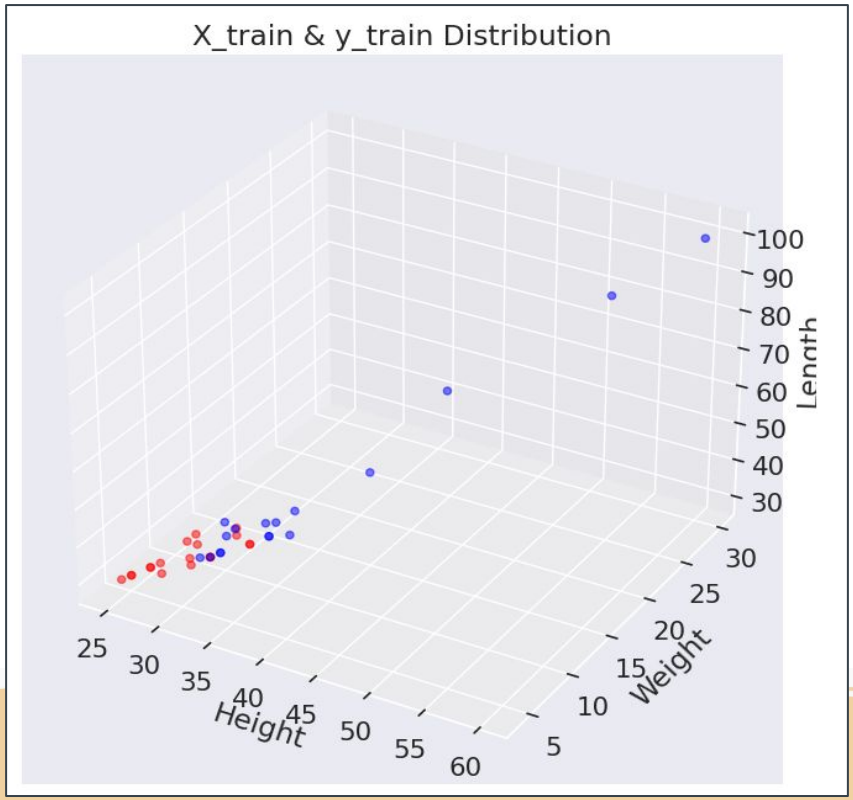
1. Split our data into training and testing sets with a ratio of 8:2.

```
X = df[['Height', 'Weight', 'Length']].values
y = df['Animal'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)
```

2. Initialize and fit the model.

```
#Initalization
my_knn = KNN(k=3, metric="euclidean")

#Fit model
my_knn.fit(X_train, y_train)
```

X_train and y_train Distribution

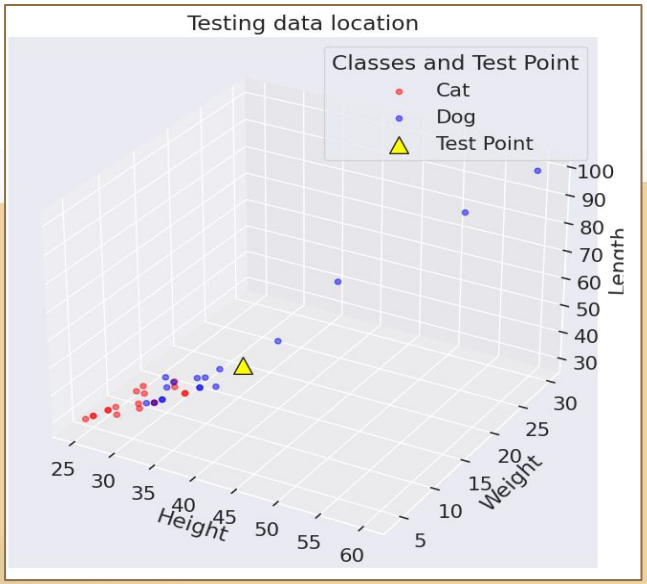


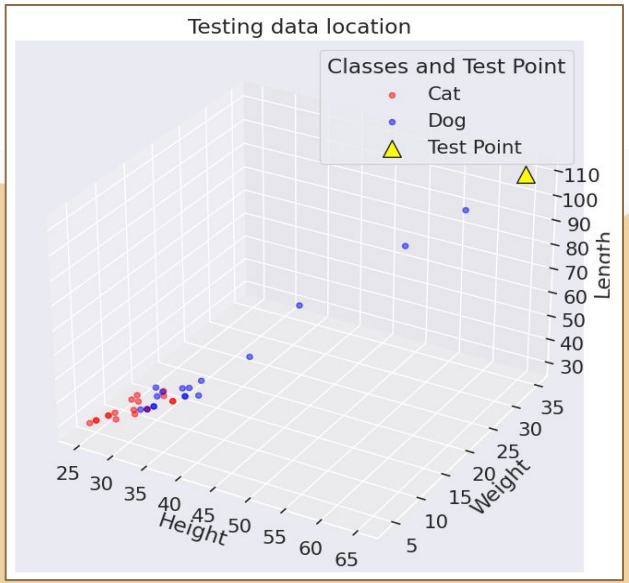
KNN Model Working Visualization(Continuous)

3. Predict the label by X_test after our KNN model training completely.

```
#Prediction
preds  =  my_knn.predict(X_test)
```

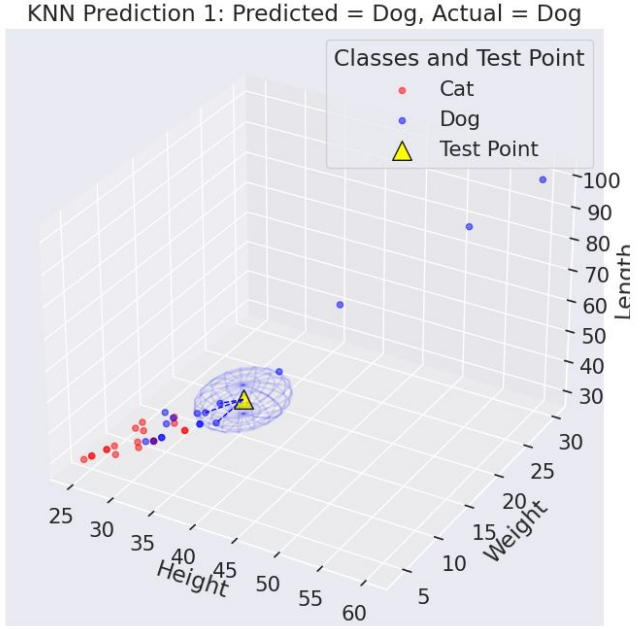
Every X_test data will calculate the distances with all X_train data based on the metrics we selected.

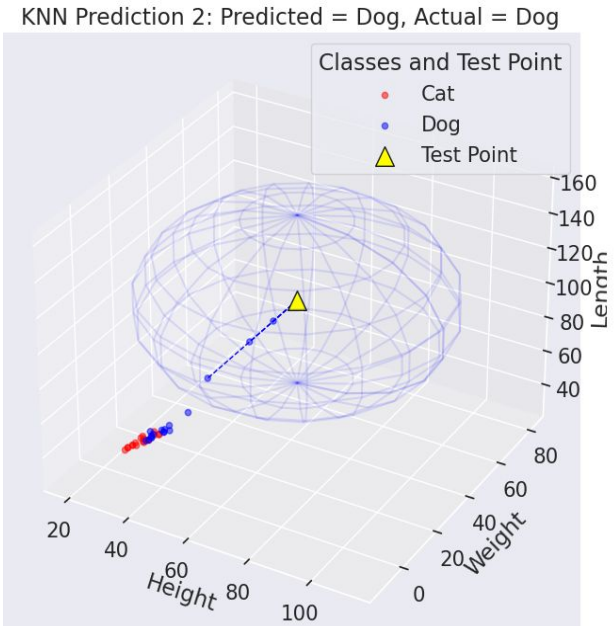


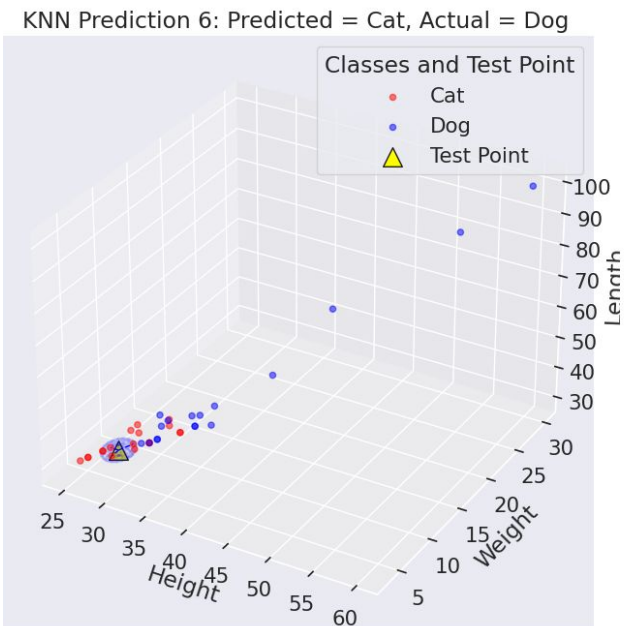


KNN Model Working Visualization(Continuous)

- After calculating all the distance between each X_test row with all X_train data, we got a distance list and sort them with distance and return top k nearest point.
- Compare the prediction label with true y_label to check if the result is right.







KNN Model Working Visualization(Continuous)

4. Based on the prediction label and true y_test label, we get score of our model.

```
#check accuracy
accuracy = my_knn.score(preds, y_test)
print(f"Accuracy: {accuracy:.2f}%")
```

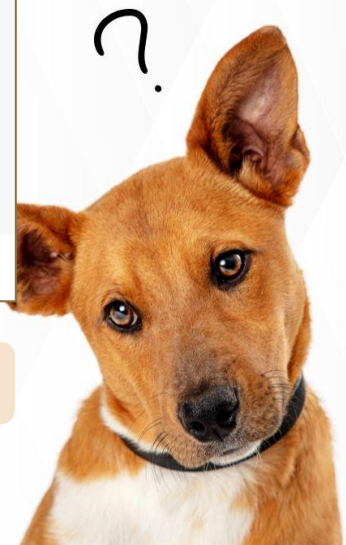
```
Accuracy: 55.56%
```

Our KNN vs Official KNN

```
#standard KNN from sklearn
# Initialize the KNN classifier with k=3 (3 nearest neighbors)
knn_model = KNeighborsClassifier(n_neighbors=3, metric="euclidean")
# Fit the KNN model using the training data (X_train, y_train)
knn_model.fit(X_train,y_train)
# Predict the class labels for the test set (X_test)
preds = knn_model.predict(X_test)

# Evaluate the model by calculating the accuracy score on the test data
accuracy = knn_model.score(X_test, y_test)
print(f"{round(accuracy*100,2)}%")
```

```
55.56%
```

We compare the accuracy result between our scratch KNN and standard KNN from sklearn package



Conclusion

KNN is a simple and efficient algorithm for classification tasks.

Identified a portion of the cats and dogs in the dataset, still room for improvement.

Our custom KNN model achieves similar accuracy and performance compared to the standard KNN implementation in scikit-learn on this dataset.

Challenges

1 Experimenting with Different Values for 'k'

Finding the best number of neighbors 'k' for the model is tricky. Using too few or too many neighbors can affect how well the model works.

2 Exploring Other Distance Metrics

The way we measure distance between data points affects our model. While we often use Euclidean distance, other types like Manhattan distance might work better.

3 Applying the Model to Datasets with clear boundary

From previous dataset distribution, we can see that this dataset does not exhibit strong clustering characteristics, and the classification boundaries between data points are not clear, which limits the KNN model's ability to achieve good performance during testing.

Citation

1. CatsAndDogs dataset from Kaggle: https://www.kaggle.com/datasets/scarb7/catsanddogs-dummy
2. https://www.kaggle.com/code/samuelcortinhas/k-nearest-neighbours-knn-from-scratch
3. https://www.geeksforgeeks.org/k-nearest-neighbours

Before Presentation

After Presentation

Questions