Predicting Categories: Getting Started with Classification

Predicting Categories: Getting Started with Classification

- Classification Tasks
- A Simple Classification Dataset
- Training and Testing: Don’t Teach to the Test
- Evaluation: Grading the Exam
- Simple Classifier #1: Nearest Neighbors, Long Distance Relationships, and Assumptions

3.1 Classification Tasks

- Classification is used to identify the category of new observations on the basis of training data.
- Binary classification refers to predicting one of two classes.
- Multi-class classification involves predicting one of more than two classes.
- Some classifiers try to make a decision about the output in a direct fashion.
- Classifiers break the decision into a two-step process:
    - ✓ Build a model of how likely the outcomes are.
    - ✓ Pick the most likely outcome.

3.2 A Simple Classification Dataset

- The iris dataset is included with **sklearn** and it has a long, rich history in machine learning and statistics.
- It is also called Fisher's Iris Dataset because Sir Ronald Fisher, a statistician, used it as the sample data in one of the first academic papers.
- A histogram is a good way to view the distribution of a continuous numeric variable.
- The **hist()** method is used to draw the histogram.
- **sns.pairplot** gives us a nice panel of graphics.
- It is used to get the relation between each and every variable present in Pandas data frame.

3.3 Training and Testing: Don’t Teach to the Test

- A teach-to-the-test evaluation scheme is called an in-sample evaluation or training error.
- The **train_test_split** function segments our dataset that lives in the Python.
- It is used for splitting data arrays into two subsets: for training data and for testing data.

3.4 Evaluation: Grading the Exam

» Evaluation is the subsidiary part of the model development process.

» This phase decides whether the model performs better.

» Accuracy determines which model is best at identifying relationships and patterns between variables in a dataset.

» It is based on the input or training data.

» Accuracy can be calculated by hand in three steps:

- ✓ Mark each answer right or wrong.
- ✓ Add up the correct answers.
- ✓ Calculate the percent.

3.4 Evaluation: Grading the Exam (continued)

» **sklearn's metrics.accuracy_score** is used to calculate accuracy by coding.

» Here's an example:

```
print("sklearn accuracy:",
      metrics.accuracy_score(answer_key,
                             student_answers))
```

3.5 Nearest Neighbors, Long Distance Relationships, and Assumptions

- Defining Similarity
- The k in k-NN
- Parameters, Nonparametric Methods & Hyperparameter
- Building a k-NN Classification Model

3.5.2 The k-NN

» The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm.

» It can be used to solve both classification and regression problems.

» K represents the number of nearest neighbors you want to select for making prediction.

- Classification:

Votes and assigns the K Nearest Neighbors Class label to the unknown data point

- Regression:

Take average value of its K Nearest Neighbors and assigns to the unknown data point.

» Here are the ideas for making predictions from a labeled dataset:

- ✓ Find a way to describe the similarity of two different examples.
- ✓ When you need to make a prediction on a new, unknown example, simply take the value from the most similar known example.

» Similarity can be defined by calculating a distance between pairs of examples.

- ✓ $similarity = distance(example\_one, example\_two)$

3.5.3 Answer Combination

» If we have an animal classification problem, four of our nearest neighbors might vote for cat, cat, dog, and zebra.

» How do we respond for our test example?

» It seems like taking the most frequent response, cat, would be a decent method.

» We can use the exact same neighbor-based technique in regression problems where we try to predict a numerical value.

» The only thing we have to change is how we combine our neighbors' targets.

3.5.3 Answer Combination

- » If three of our nearest neighbors gave us numerical values of 3.1, 2.2, and 7.1, how do we combine them?
- » We could use any statistic we wanted, but the mean (average) and the median (middle) are two common and useful choices.

3.5.4 k-NN, Parameters, and Nonparametric Methods

» k-NN outputs (the predictions) can't be computed from an input example.

» k-NN is a **nonparametric** learning method.

» It means the relationship between features and targets cannot be captured solely using a fixed number of parameters.

3.5.4 k-NN, Parameters, and Nonparametric Methods

Parameters are the internal configurations of a model that are learned from the training data. They are specific to the model and are optimized through the training process.

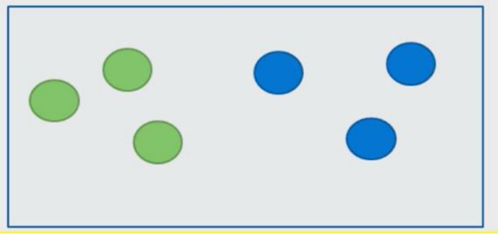
- **Examples**: In a neural network, weights and biases are parameters. In a linear regression model, the slope and intercept of the regression line are parameters.

Nonparametric models are a class of statistical models that do not assume a specific form or finite set of parameters for the underlying data distribution. Instead, they are more flexible and can grow in complexity as more data becomes available.

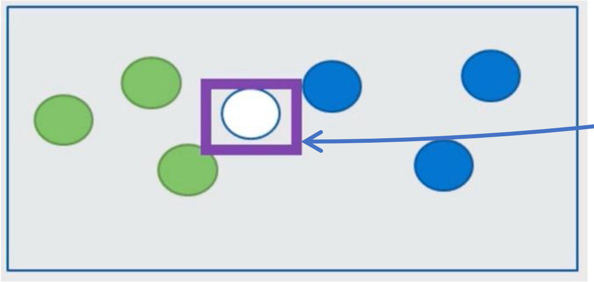
Hyperparameters are external configurations that are set before the training process begins. They control the learning process but are not learned from the data.

Example (Lazy Learner)

Training: It just store the training data in memory and does not learn any patterns. Due to this training will be very fast.



Prediction: It calculates the distance between the **unknown point** and all other training points and make predictions on K Nearest Neighbours class label for classification and KNN average value for Regression.



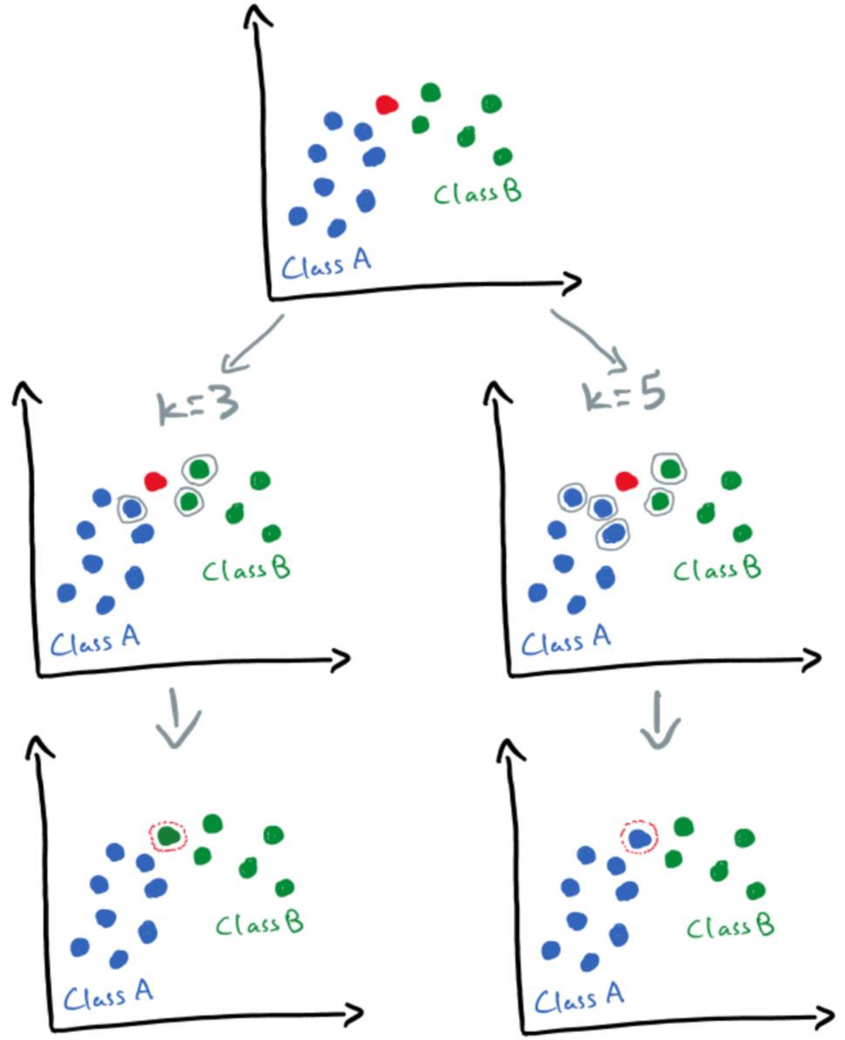
Example

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

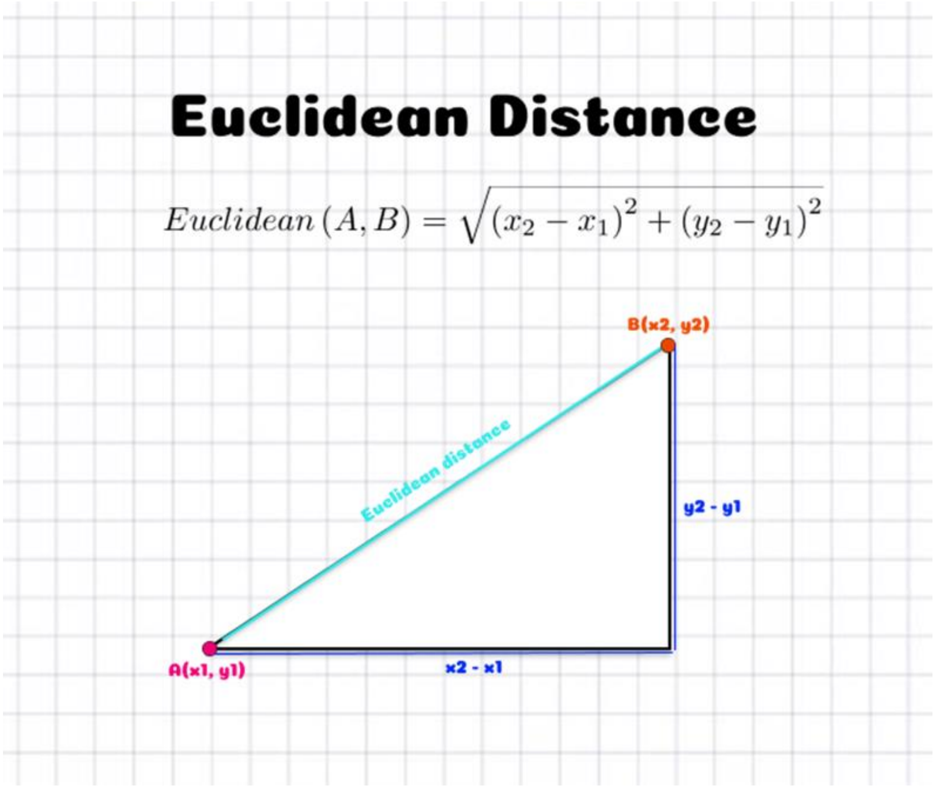
Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

Distance Functions



Distance Function



Two, Three and n dimensions - Euclidean Distance

- if **p** = (p_1, p_2) and **q** = (q_1, q_2) then the distance is given by

$$d(p,q) = \sqrt{(p1 = q1)^2 + (p2 - q2)^2}$$

- In three-dimensional Euclidean space, the distance is

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}.$$

- In general, for an *n*-dimensional space, the distance is

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}.$$

Example

Training set

ex_1	{[1,3,1],pos}
ex_2	{[3,5,2],pos}
ex_3	{[3,2,2],neg}
ex_4	{[5,2,3],neg}

Scenario Description:

• We have a training set with four examples, each having three numeric attributes.

• Our goal is to classify a new data point, 'x,' which has the attribute values [2, 4, 2].

Solution:

Step 1: Calculating Euclidean Distances

• First, we calculate the Euclidean distances between 'x' and all the training examples.

• Euclidean distance measures the 'closeness' between data points in a multi-dimensional space."

		Distance between ex_i and [2,4,2]
ex_1	{[1,3,1],pos}	$\sqrt{(2-1)^2+(4-3)^2+(2-1)^2}=\sqrt{3}$
ex_2	{[3,5,2],pos}	$\sqrt{(2-3)^2+(4-5)^2+(2-2)^2}=\sqrt{2}$
ex_3	{[3,2,2],neg}	$\sqrt{(2-3)^2+(4-2)^2+(2-2)^2}=\sqrt{5}$
ex_4	{[5,2,3],neg}	$\sqrt{(2-5)^2+(4-2)^2+(2-3)^2}=\sqrt{14}$

Example

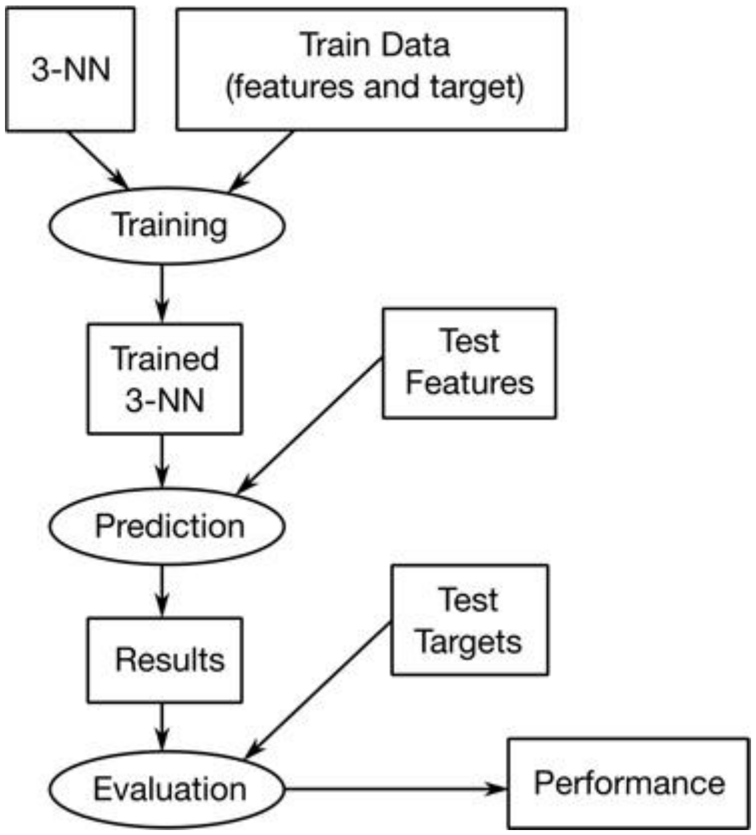
Step 2: Identifying Nearest Neighbor
• Upon calculating the distances, we find that 'x's nearest neighbor is 'ex2.'"
• This means that 'ex2' in our training set is the most similar data point to 'x'based on the attribute values.
Step 3: k-NN Classification (k=1)
• Now, we perform a 1-NN classification, where 'k' is set to 1.
• Since 'ex2' is the nearest neighbor and its label is 'pos' (positive), the 1-NNclassifier returns the positive label for 'x'.

What class does 'x' belong to when 'k' equals 3?

• In this case, 'ex1,' 'ex2,' and 'ex3' are the three nearest neighbors to 'x.'
• Among these neighbors, 'ex1' and 'ex2' are positive, while 'ex3' is negative.

3.5.5 Building a k-NN Classification Model

» Here's a workflow of training, testing, and evaluation for 3-NN:



3.5.5 Building a k-NN Classification Model (continued)

» Here are the steps for creating and estimating a 3-NN model:

- ✓ Create a 3-NN model.
- ✓ Fit that model on the training data.
- ✓ Use that model to predict on the test data.
- ✓ Evaluate those predictions using accuracy.

3.5.5 Building a k-NN Classification Model (IRIS Dataset)

```
import pandas as pd
from sklearn import datasets
import numpy as np
import seaborn as sns
import sklearn.model_selection as skms
import sklearn.neighbors as neighbors
import sklearn.metrics as metrics
```

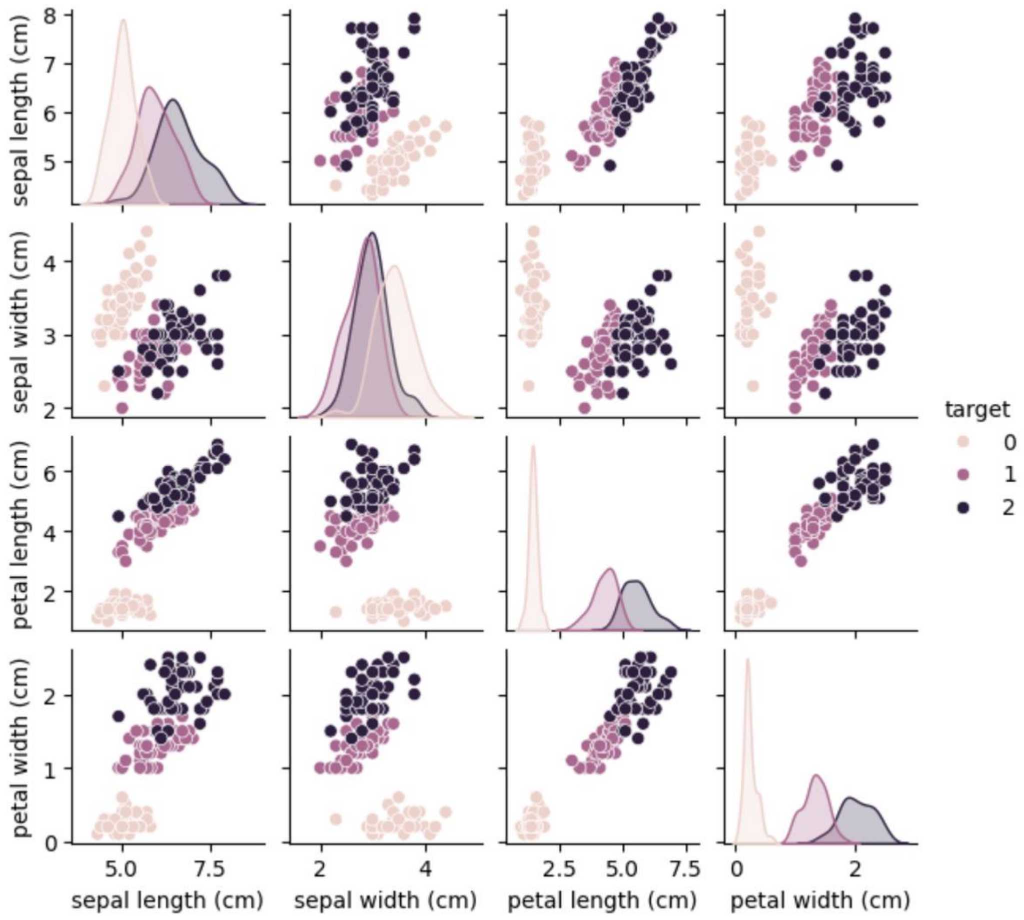
3.5.5 Building a k-NN Classification Model (IRIS Dataset)

```
iris = datasets.load_iris()
irisdf = pd.DataFrame(iris.data, columns = iris.feature_names)
irisdf['target'] = iris.target
irisdf.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

3.5.5 Building a k-NN Classification Model (IRIS Dataset)

```
sns.pairplot(irisdf, hue = 'target', size=1.5)
```



3.5.5 Building a k-NN Classification Model (IRIS Dataset)

```
print('targets: {}'.format(iris.target_names), iris.target_names[0], sep='\n')
```

```
targets: ['setosa' 'versicolor' 'virginica'] setosa
```

```
(iris_train_ftrs, iris_test_ftrs, iris_train_tgt, iris_test_tgt)
= skms.train_test_split(iris.data, iris.target, test_size=0.25)

print("Train feature shape : ", iris_train_ftrs.shape)
print("Test feature shape : ", iris_test_ftrs.shape)
```

```
Train feature shape : (112, 4)
Test feature shape : (38, 4)
```

3.5.5 Building a k-NN Classification Model (IRIS Dataset)

```
knn = neighbors.KNeighborsClassifier(n_neighbors=3)

fit = knn.fit(iris_train_ftrs, iris_train_tgt)

preds=fit.predict(iris_test_ftrs)

print('3NN Accuracy = ', metrics.accuracy_score(iris_test_tgt, preds))
```

```
3NN Accuracy = 0.9736842105263158
```

3.5.5 Building a k-NN Classification Model (IRIS Dataset)

```
# Get user input for features
predicted_target_index = knn.predict([[20,12,13,4]])


# Predict the target
predicted_target_name =
iris.target_names[predicted_target_index[0]]


print(predicted_target_name)
```

```
virginica
```

```python
# Get user input for features
sl = float(input("Enter sepal length (cm): "))
sw = float(input("Enter sepal width (cm): "))
pl = float(input("Enter petal length (cm): "))
pw = float(input("Enter petal width (cm): "))

# The model expects a 2D array, even for a single sample.
user_input_features = np.array([[sl, sw, pl, pw]])
# Make a prediction using the trained model
predicted_target_index = knn.predict(user_input_features)
# Get the actual target name
predicted_target_name = iris.target_names[predicted_target_index[0]]
print(predicted_target_index, " = ", predicted_target_name)
```

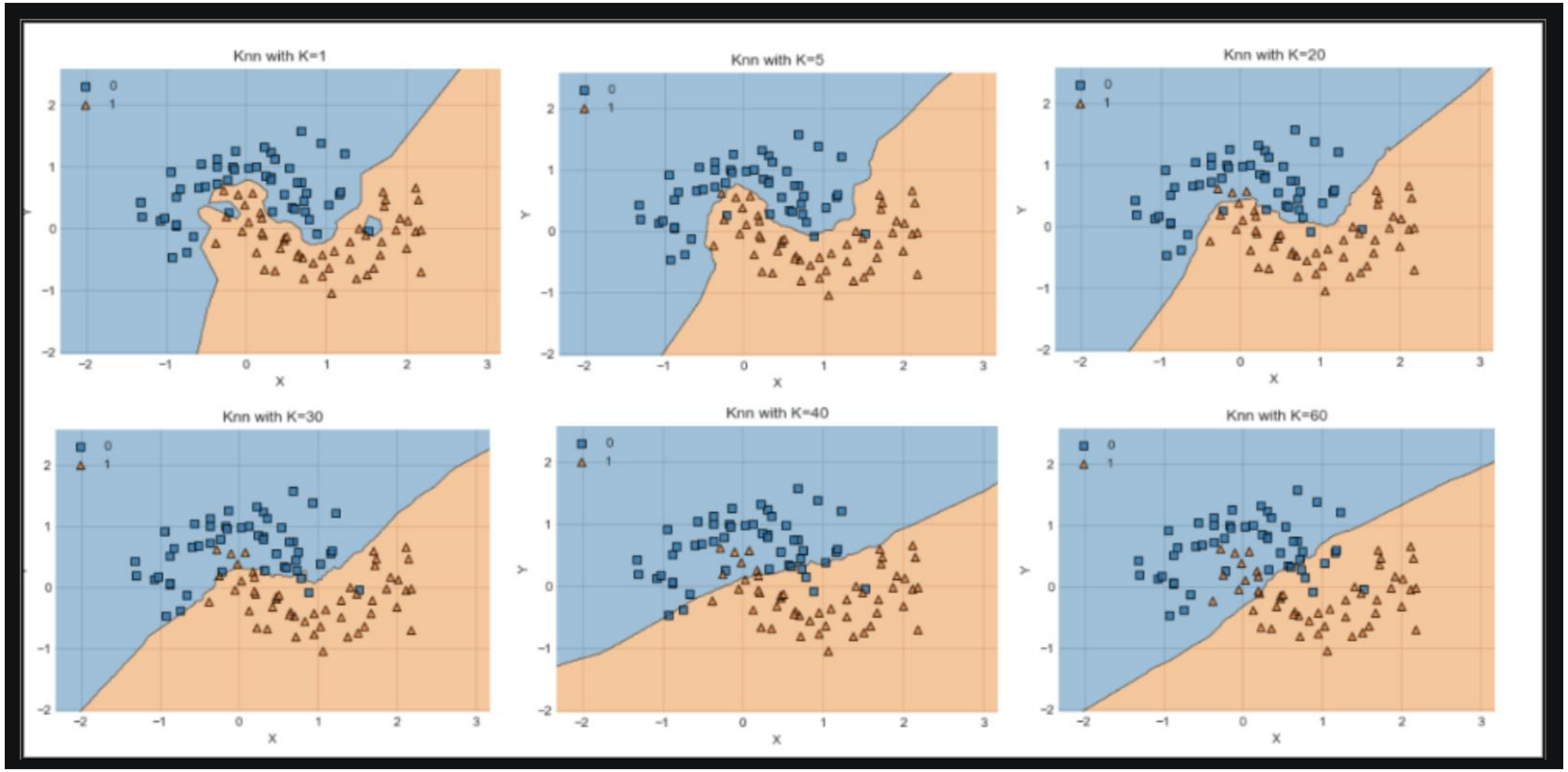
```
Enter sepal length (cm): 2.5
Enter sepal width (cm): 12.4
Enter petal length (cm): 2.7
Enter petal width (cm): 5.4
[0] = setosa
```

Choosing the best value for K

- To find the optimal K value for your data, you'll typically run the KNN algorithm multiple times with varying K values and evaluate each scenario based on accuracy. If the accuracy is stable as K changes, that K value may be a suitable choice.
- When selecting K, consider that the feature count and group size are influential factors in the model's performance. More features or more classes often require larger K values to capture meaningful patterns in the data.

Choosing the best value for K

- **Higher K values:** Increasing K generally stabilizes predictions and improves resilience to outliers. A practical approach is incrementally increasing K until your chosen accuracy metric meets an acceptable threshold.
- **K = 1:** This makes predictions highly sensitive to noise and outliers



Clustering results with different K values. Increasing K results in a more granular segmentation that captures finer details (see the first row). However, as K approaches the number of observations (last scenario), the segmentation becomes less effective, leading to poor results and overfitting.

Pros & Cons

Pros

- Useful for nonlinear data because KNN is a nonparametric algorithm.
- Can be used for both classification and regression problems, even though mostly used for classification.

Cons

- Difficult to choose *K* since there is no statistical way to determine that.
- Slow prediction for large datasets.
- Computationally expensive since it has to store all the training data (Lazy Learner).
- Sensitive to non-normalized dataset.
- Sensitive to presence of irrelevant features.