LECTURE : 08

KNN K NEAREST NEIGHBOR

WHAT IS KNN?

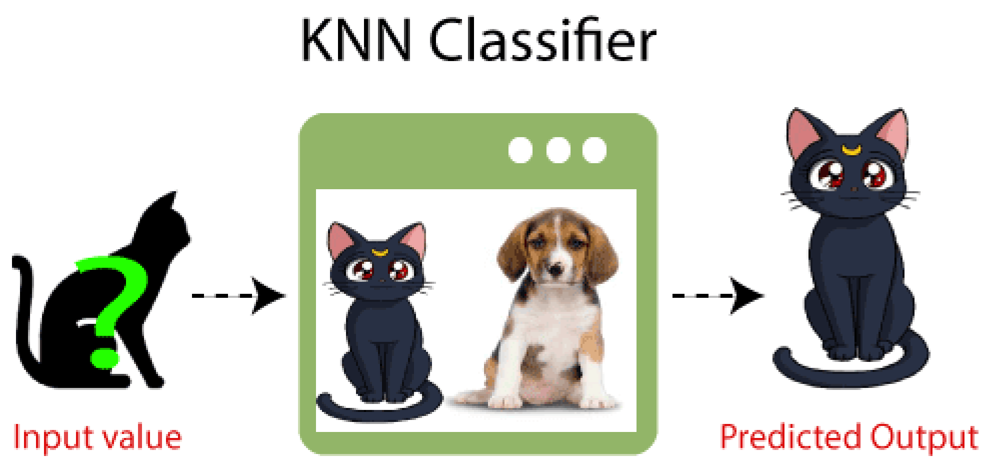
- K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

CONTINUE:

- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

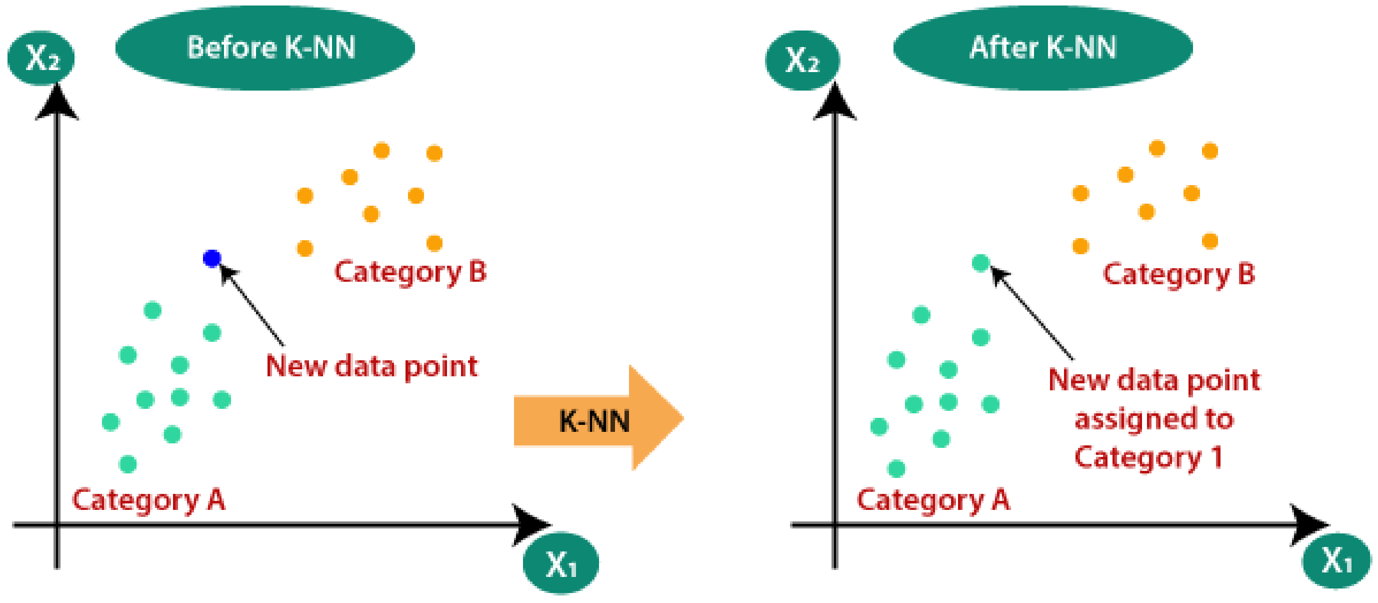
EXAMPLE:

- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So, for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs' images and based on the most similar features it will put it in either cat or dog category.



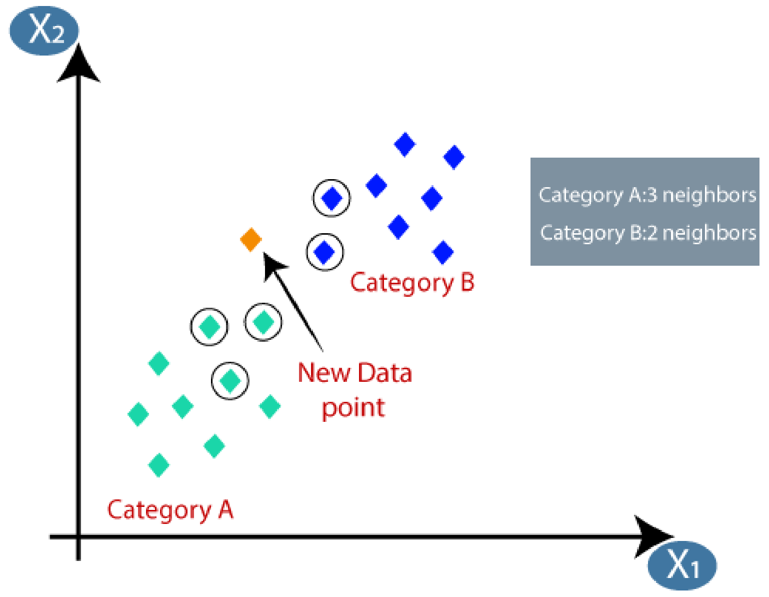
WHY DO WE NEED KNN?

- Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



HOW DOES KNN WORKS?

- The K-NN working can be explained on the basis of the below algorithm:
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.



HOW TO SELECT VALUE OF K IN KNN?

- Below are some points to remember while selecting the value of K in the K-NN algorithm:
- There is no way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

APPLICATIONS OF KNN ?

1. **Classification**: KNN is widely used for classification tasks in machine learning. It can classify data points into different categories based on the majority class among their nearest neighbors. Applications include email spam detection, sentiment analysis, and image recognition.
2. **Recommendation Systems**: KNN can be used in collaborative filtering-based recommendation systems to suggest items or products to users based on the preferences of similar users. For example, recommending movies or music based on the preferences of users with similar viewing or listening history.
3. **Anomaly Detection**: KNN can detect anomalies or outliers in datasets by identifying data points that are significantly different from their neighbors. This is useful in fraud detection, network intrusion detection, and manufacturing quality control.

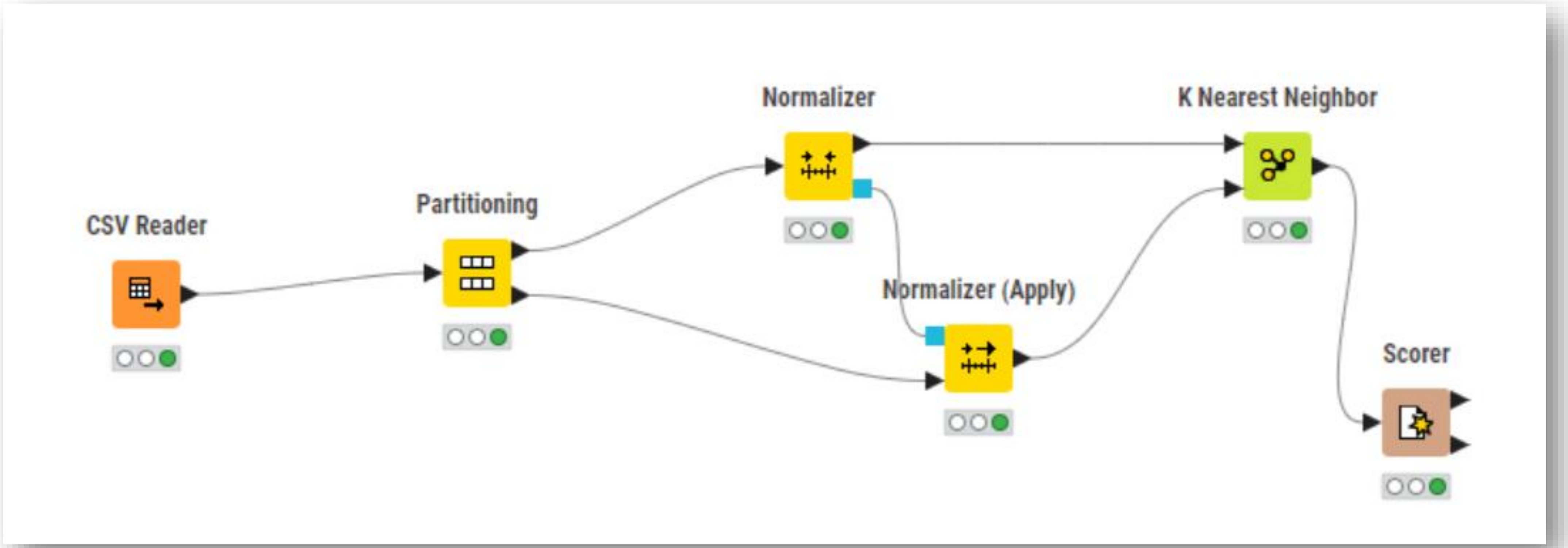
ADVANTAGES OF KNN?

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

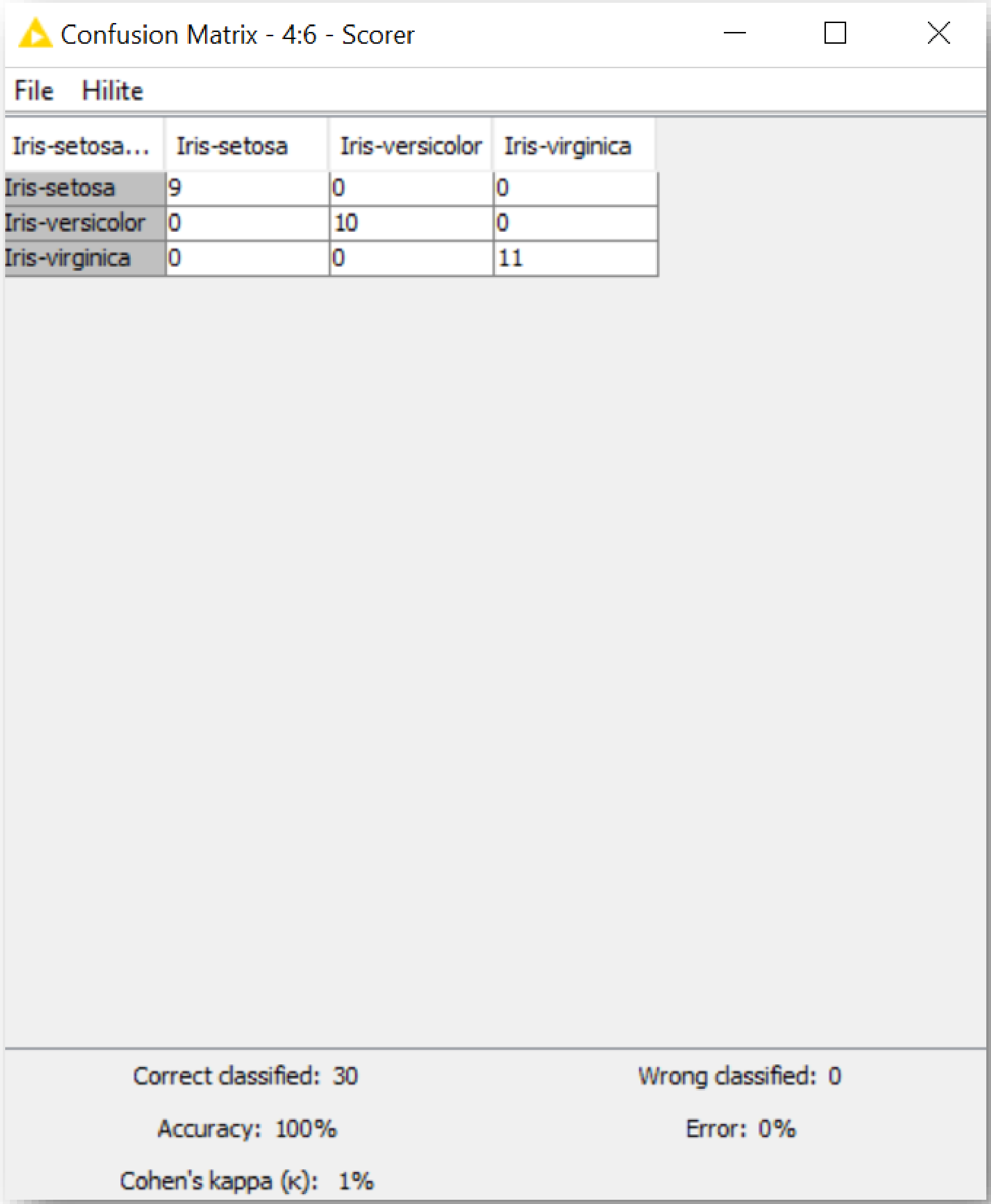
DISADVANATGES OF KNN?

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

KNN IN KNIME?


CSV Reader
Partitioning
Normalizer
Normalizer (Apply)
K Nearest Neighbor
Scorer

ACCURACY RATE:



Iris-setosa...	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	9	0	0
Iris-versicolor	0	10	0
Iris-virginica	0	0	11

Correct classified: 30 Wrong classified: 0

Accuracy: 100% Error: 0%

Cohen's kappa (κ): 1%

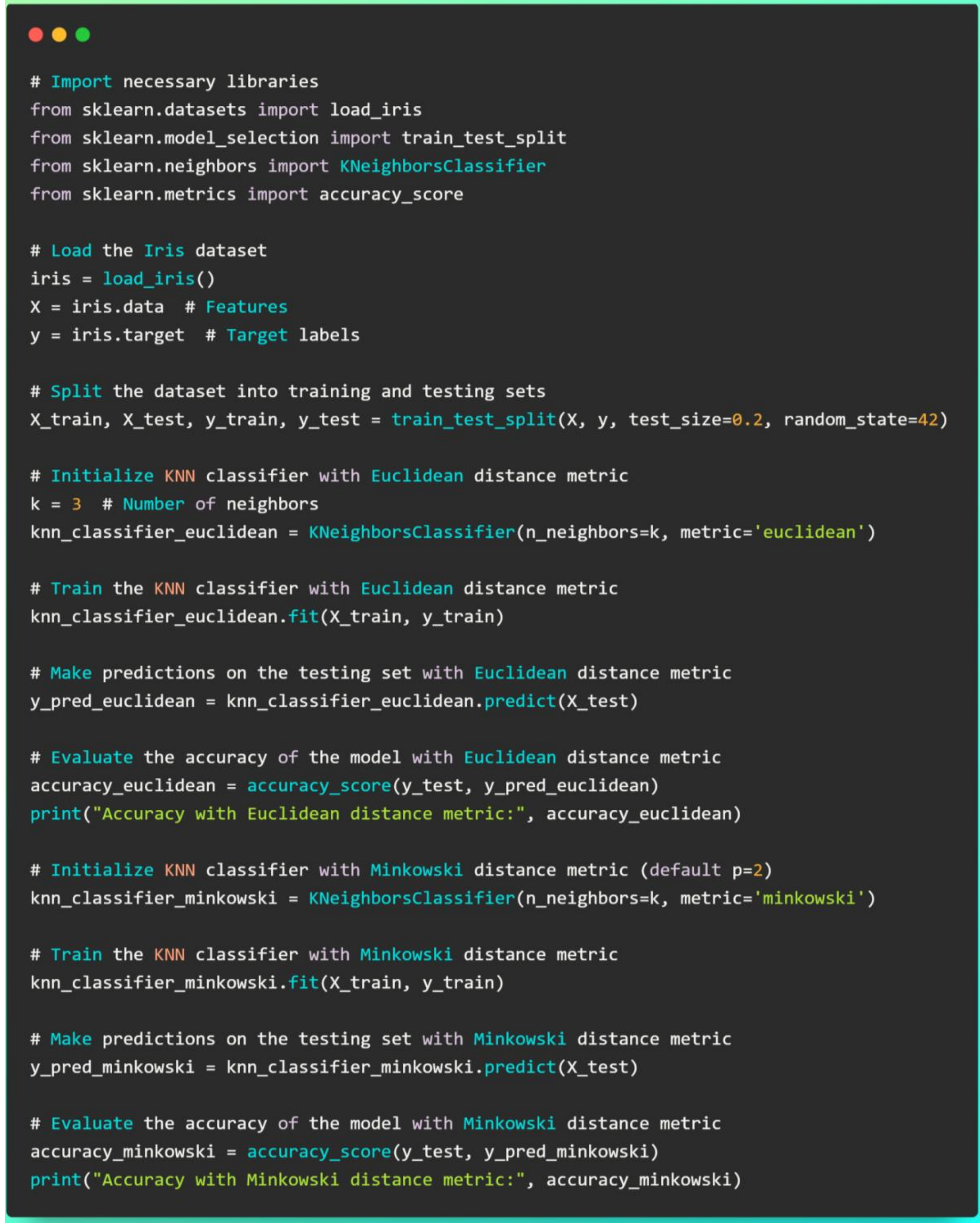
STEPS TO PERFORM KNN:

- Data Pre-processing step
- Fitting the K-NN algorithm to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result.

METRICS IN KNN?

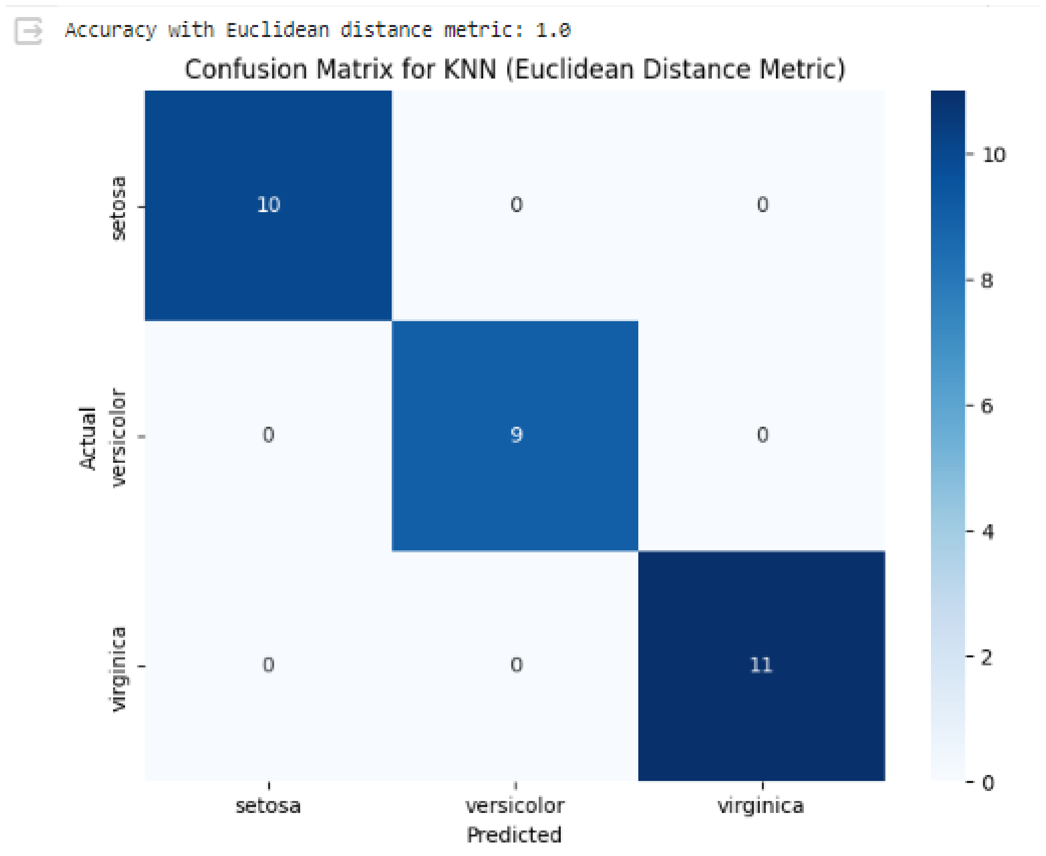
- In K-Nearest Neighbors (KNN), several distance metrics are commonly used to measure the similarity between data points and determine the nearest neighbors. The choice of distance metric can significantly impact the performance of the KNN algorithm.

KNN IN PYTHON:

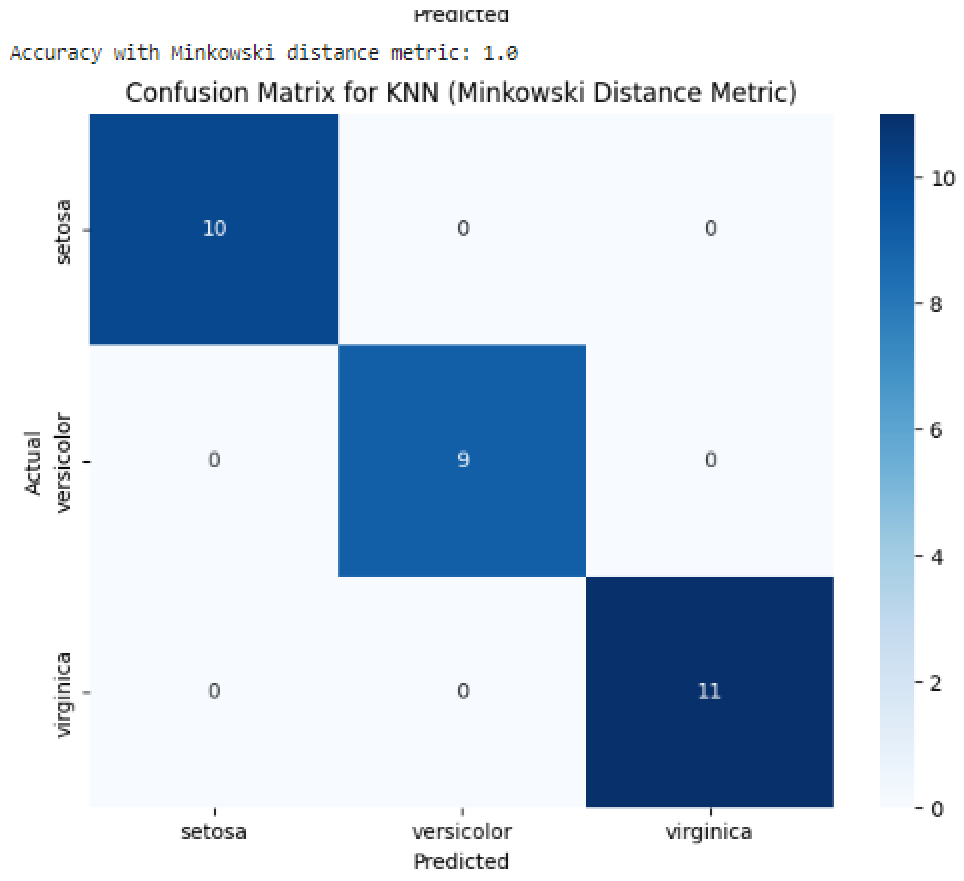


```python
# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
X = iris.data  # Features
y = iris.target  # Target labels

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize KNN classifier with Euclidean distance metric
k = 3  # Number of neighbors
knn_classifier_euclidean = KNeighborsClassifier(n_neighbors=k, metric='euclidean')

# Train the KNN classifier with Euclidean distance metric
knn_classifier_euclidean.fit(X_train, y_train)

# Make predictions on the testing set with Euclidean distance metric
y_pred_euclidean = knn_classifier_euclidean.predict(X_test)

# Evaluate the accuracy of the model with Euclidean distance metric
accuracy_euclidean = accuracy_score(y_test, y_pred_euclidean)
print("Accuracy with Euclidean distance metric:", accuracy_euclidean)

# Initialize KNN classifier with Minkowski distance metric (default p=2)
knn_classifier_minkowski = KNeighborsClassifier(n_neighbors=k, metric='minkowski')

# Train the KNN classifier with Minkowski distance metric
knn_classifier_minkowski.fit(X_train, y_train)

# Make predictions on the testing set with Minkowski distance metric
y_pred_minkowski = knn_classifier_minkowski.predict(X_test)

# Evaluate the accuracy of the model with Minkowski distance metric
accuracy_minkowski = accuracy_score(y_test, y_pred_minkowski)
print("Accuracy with Minkowski distance metric:", accuracy_minkowski)
```

RESULTS:



```
Accuracy with Euclidean distance metric: 1.0
```



```
Accuracy with Minkowski distance metric: 1.0
```

	setosa	versicolor	virginica
setosa	10	0	0
versicolor	0	9	0
virginica	0	0	11

	setosa	versicolor	virginica
setosa	10	0	0
versicolor	0	9	0
virginica	0	0	11

TASK 1:

- **Task:**
- Implement KNN for Iris Flower Classification in KNIME

Description: You are tasked with implementing the K-Nearest Neighbors (KNN) algorithm using KNIME Analytics Platform to classify Iris flowers based on their features. The Iris dataset contains four features: sepal length, sepal width, petal length, and petal width. There are three classes of Iris flowers in the dataset: Setosa, Versicolor, and Virginica.

- **Steps:**

1. Load the Iris dataset into KNIME.
2. Preprocess the data as necessary, including handling missing values, if any.
3. Split the dataset into training and testing sets using the Split Data node.
4. Configure the KNN Learner node to train the KNN classifier with a chosen number of neighbors and distance metric.
5. Use the trained KNN model to make predictions on the testing set.
6. Evaluate the performance of the classifier using the Scorer node to calculate accuracy.
7. Optionally, visualize the results using appropriate nodes such as the Interactive Table or Confusion Matrix.

TASK 2:

- **Task:**
- Implement KNN for Iris Flower Classification
- **Description:** You are tasked with implementing the K-Nearest Neighbors (KNN) algorithm to classify Iris flowers based on their features. The Iris dataset contains four features: sepal length, sepal width, petal length, and petal width. There are three classes of Iris flowers in the dataset: Setosa, Versicolor, and Virginica.

- **Steps:**

1. Load the Iris dataset using scikit-learn.
2. Split the dataset into training and testing sets.
3. Initialize the KNN classifier with a chosen number of neighbors.
4. Train the KNN classifier using the training data.
5. Make predictions on the testing set using the trained classifier with different metrics.
6. Evaluate the performance of the classifier using accuracy score and confusion matrix.
7. Visualize the results using appropriate plots or visualizations.

TASK 3:

- **Task:**
- Implement a Movie Recommendation System Using K-Nearest Neighbors (KNN) in KNIME
- **Description:** You are tasked with building a movie recommendation system using the K-Nearest Neighbors (KNN) algorithm in KNIME Analytics Platform. The goal of the recommendation system is to suggest movies to users based on their similarity to other users' preferences.
- **Steps:**

1. Load the movie ratings dataset into KNIME. The dataset should contain information about users, movies, and their ratings.
2. Preprocess the data as necessary, including handling missing values and encoding categorical variables.
3. Construct a user-item matrix where rows represent users, columns represent movies, and each cell contains the rating given by the user to the movie.
4. Implement the KNN algorithm to find similar users based on their movie preferences. You may use distance metrics such as cosine similarity or Pearson correlation coefficient.
5. Choose a suitable number of neighbors (k) for the KNN algorithm.
6. Recommend movies to a target user by selecting movies that similar users have rated highly but the target user has not yet seen.
7. Evaluate the performance of the recommendation system using appropriate metrics, precision, recall, or mean average precision.