K-Nearest Neighbors (KNN) Classifier and Imputation using KNN

Bindeshwar Singh Kushwaha

What is K-Nearest Neighbors (KNN)?

- KNN is a supervised machine learning algorithm.

What is K-Nearest Neighbors (KNN)?

- KNN is a supervised machine learning algorithm.
- It is easy to understand and does not involve complex math.

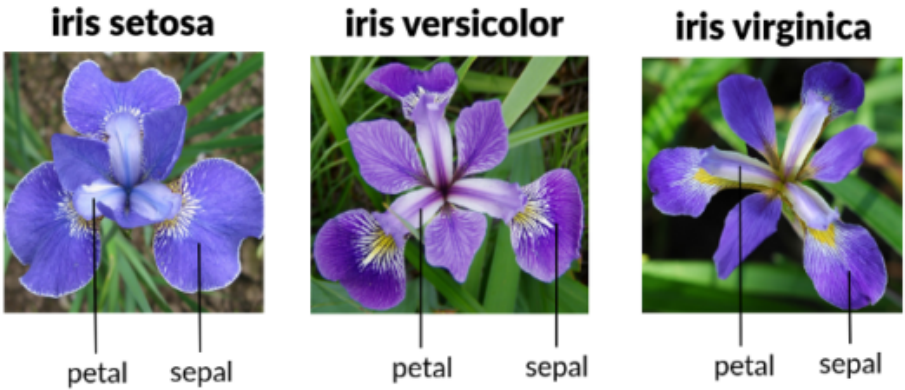
What is K-Nearest Neighbors (KNN)?

- KNN is a supervised machine learning algorithm.
- It is easy to understand and does not involve complex math.
- Commonly used for classification tasks, especially with labeled data.

What is K-Nearest Neighbors (KNN)?

- KNN is a supervised machine learning algorithm.
- It is easy to understand and does not involve complex math.
- Commonly used for classification tasks, especially with labeled data.
- We'll use the Iris dataset, which has flower measurements.

Iris Flower Types



Iris Dataset Sample Features

- The dataset includes four features:

Iris Dataset Sample Features

- The dataset includes four features:
  - Sepal Length

Iris Dataset Sample Features

- The dataset includes four features:
  - Sepal Length
  - Sepal Width

Iris Dataset Sample Features

- The dataset includes four features:
  - Sepal Length
  - Sepal Width
  - Petal Length

Iris Dataset Sample Features

- The dataset includes four features:
  - Sepal Length
  - Sepal Width
  - Petal Length
  - Petal Width

Iris Dataset Sample Features

- The dataset includes four features:
  - Sepal Length
  - Sepal Width
  - Petal Length
  - Petal Width
- Target class is the Iris flower type: Setosa, Versicolor, or Virginica.

Sample Data Points

In original dataset there are 150 instances which has four features sepal length, sepal width, petal length and petal width and three classes Iris-setosa, Iris-versicolor and Irish-virginica. For the sake of simplicity, I have only take only ten instances in which nine are labeled and one is not labeled. Nine instances will be used for training and one will be used for testing.

S. No.	Sepal Len. (cm)	Sepal Width (cm)	Petal Len. (cm)	Petal Width (cm)	Label
1	4.6	3.4	1.4	0.3	Iris-setosa
2	5.0	3.4	1.5	0.2	Iris-setosa
3	4.4	2.9	1.4	0.2	Iris-setosa
4	7.0	3.2	4.7	1.4	Iris-versicolor
5	6.4	3.2	4.5	1.5	Iris-versicolor
6	6.9	3.1	4.9	1.5	Iris-versicolor
7	6.3	3.3	6.0	2.5	Iris-virginica
8	5.8	2.7	5.1	1.9	Iris-virginica
9	7.1	3.0	5.9	2.1	Iris-virginica
10	5.0	3.3	1.4	0.2	Unknown

Understanding KNN with Distance Calculation

From the data set you can see nine instances having labels and the tenth has not a label. To train the model, distance will be calculated from instance 10 to all other instances, and the value of k should be chosen.

$$d(1,10) = \sqrt{(sl_1 - sl_{10})^2 + (sw_1 - sw_{10})^2 + (pl_1 - pl_{10})^2 + (pw_1 - pw_{10})^2}$$

$$d(1,10) = \sqrt{(4.6 - 5.0)^2 + (3.4 - 3.3)^2 + (1.4 - 1.4)^2 + (0.3 - 0.2)^2} = 0.424$$

In the above equation

- sl: Sepal Length,
- sw: Sepal Width,
- pl: Petal Length,
- pw: Petal Width.

Distances from Instance 10

The same way, other distances are calculated as:

$$
\begin{aligned}
d(2,10) &= 0.141 \\
d(3,10) &= 0.721 \\
d(4,10) &= 4.042 \\
d(5,10) &= 3.643 \\
d(6,10) &= 4.194 \\
d(7,10) &= 5.305 \\
d(8,10) &= 4.193 \\
d(9,10) &= 5.325 \\
d(10,10) &= 0
\end{aligned}
$$

Conclusion with k = 3

If we take $k = 3$, the three nearest distances are:

- 0.141 (Iris-Setosa),
- 0.424 (Iris-Setosa),
- 0.721 (Iris-Setosa).

So, if $k = 3$, the majority of Iris-Setosa instances are nearest to instance number 10. Hence, we can classify instance 10 as **Iris-Setosa**.

KNN Classifier with Iris Dataset

- Import the KNN model from sklearn: `from sklearn.neighbors import KNeighborsClassifier`

KNN Classifier with Iris Dataset

- Import the KNN model from sklearn: `from sklearn.neighbors import KNeighborsClassifier`
- Create 9 training samples (`X_train`) with features like sepal and petal measurements.

KNN Classifier with Iris Dataset

- Import the KNN model from sklearn: `from sklearn.neighbors import KNeighborsClassifier`
- Create 9 training samples (`X_train`) with features like sepal and petal measurements.
- Assign class labels: Iris-setosa, Iris-versicolor, Iris-virginica (`y_train`).

KNN Classifier with Iris Dataset

- Import the KNN model from sklearn: `from sklearn.neighbors import KNeighborsClassifier`
- Create 9 training samples (`X_train`) with features like sepal and petal measurements.
- Assign class labels: Iris-setosa, Iris-versicolor, Iris-virginica (`y_train`).
- Initialize the model with 3 neighbors and Euclidean distance (`p=2`).

KNN Classifier with Iris Dataset

- Import the KNN model from sklearn: `from sklearn.neighbors import KNeighborsClassifier`
- Create 9 training samples (`X_train`) with features like sepal and petal measurements.
- Assign class labels: Iris-setosa, Iris-versicolor, Iris-virginica (`y_train`).
- Initialize the model with 3 neighbors and Euclidean distance (`p=2`).
- Fit the model using: `modal.fit(X_train, y_train)`

KNN Classifier with Iris Dataset

- Import the KNN model from sklearn: `from sklearn.neighbors import KNeighborsClassifier`
- Create 9 training samples (`X_train`) with features like sepal and petal measurements.
- Assign class labels: Iris-setosa, Iris-versicolor, Iris-virginica (`y_train`).
- Initialize the model with 3 neighbors and Euclidean distance (`p=2`).
- Fit the model using: `modal.fit(X_train, y_train)`
- Predict the class of a new sample: `X_pred = [[5, 3.3, 1.4, 0.2]]`

KNN Classifier with Iris Dataset

- Import the KNN model from sklearn: `from sklearn.neighbors import KNeighborsClassifier`
- Create 9 training samples (`X_train`) with features like sepal and petal measurements.
- Assign class labels: Iris-setosa, Iris-versicolor, Iris-virginica (`y_train`).
- Initialize the model with 3 neighbors and Euclidean distance (`p=2`).
- Fit the model using: `modal.fit(X_train, y_train)`
- Predict the class of a new sample: `X_pred = [[5, 3.3, 1.4, 0.2]]`
- Display the result: `Predicted = modal.predict(X_pred)`

What is KNN Imputation?

- A technique to fill missing values using the k-nearest neighbors.

What is KNN Imputation?

- A technique to fill missing values using the k-nearest neighbors.
- Distance is calculated using available features (e.g., Euclidean).

What is KNN Imputation?

- A technique to fill missing values using the k-nearest neighbors.
- Distance is calculated using available features (e.g., Euclidean).
- The missing value is imputed using the average or most frequent value among neighbors.

What is KNN Imputation?

- A technique to fill missing values using the k-nearest neighbors.
- Distance is calculated using available features (e.g., Euclidean).
- The missing value is imputed using the average or most frequent value among neighbors.
- Suitable when patterns exist in similar rows.

Sample Dataset

- Consider the following data:

ID	Height (cm)	Weight (kg)	Age
1	160	55	25
2	165	60	27
3	**NaN**	62	28
4	170	65	29
5	158	54	24

Sample Dataset

- Consider the following data:

ID	Height (cm)	Weight (kg)	Age
1	160	55	25
2	165	60	27
3	**NaN**	62	28
4	170	65	29
5	158	54	24

- Row 3 is missing a value in Height.

Step-by-Step KNN Imputation (k = 2)

- We calculate distances using known values: Weight and Age.

Step-by-Step KNN Imputation (k = 2)

- We calculate distances using known values: Weight and Age.
- Row 3 = [?, 62, 28]

Step-by-Step KNN Imputation (k = 2)

- We calculate distances using known values: Weight and Age.
- Row 3 = [?, 62, 28]
- Compute distance to other rows using Weight and Age only.

Step-by-Step KNN Imputation (k = 2)

- We calculate distances using known values: Weight and Age.
- Row 3 = [?, 62, 28]
- Compute distance to other rows using Weight and Age only.
- Nearest neighbors (based on lowest distance): Rows 2 and 4

Step-by-Step KNN Imputation (k = 2)

- We calculate distances using known values: Weight and Age.
- Row 3 = [?, 62, 28]
- Compute distance to other rows using Weight and Age only.
- Nearest neighbors (based on lowest distance): Rows 2 and 4
- Heights of neighbors = 165 and 170

Step-by-Step KNN Imputation (k = 2)

- We calculate distances using known values: Weight and Age.
- Row 3 = [?, 62, 28]
- Compute distance to other rows using Weight and Age only.
- Nearest neighbors (based on lowest distance): Rows 2 and 4
- Heights of neighbors = 165 and 170
- Imputed Height = $\frac{165+170}{2} = 167.5$

Final Imputed Dataset

ID	Height (cm)	Weight (kg)	Age
1	160	55	25
2	165	60	27
3	**167.5**	62	28
4	170	65	29
5	158	54	24

- The missing height has been filled using KNN with $k = 2$.

Conclusion

- KNN Imputation is effective for datasets with structured patterns.

Conclusion

- KNN Imputation is effective for datasets with structured patterns.
- Choose an appropriate ‘k‘ based on dataset size and similarity.

Conclusion

- KNN Imputation is effective for datasets with structured patterns.
- Choose an appropriate ‘k‘ based on dataset size and similarity.
- Normalize features before applying KNN for consistent results.

KNN Imputation in Python: Step-by-Step

- Import necessary libraries: `pandas`, `numpy`, and `KNNImputer` from `sklearn`.

KNN Imputation in Python: Step-by-Step

- Import necessary libraries: `pandas`, `numpy`, and `KNNImputer` from `sklearn`.
- Create a sample dataset with a missing value in the "Height (cm)" column.

KNN Imputation in Python: Step-by-Step

- Import necessary libraries: `pandas`, `numpy`, and `KNNImputer` from `sklearn`.
- Create a sample dataset with a missing value in the "Height (cm)" column.
- Drop the "ID" column to exclude it from distance calculations.

KNN Imputation in Python: Step-by-Step

- Import necessary libraries: `pandas`, `numpy`, and `KNNImputer` from `sklearn`.
- Create a sample dataset with a missing value in the "Height (cm)" column.
- Drop the "ID" column to exclude it from distance calculations.
- Apply `KNNImputer` with `n_neighbors=2` and `metric='nan_euclidean'`.

KNN Imputation in Python: Step-by-Step

- Import necessary libraries: `pandas`, `numpy`, and `KNNImputer` from `sklearn`.
- Create a sample dataset with a missing value in the "Height (cm)" column.
- Drop the "ID" column to exclude it from distance calculations.
- Apply `KNNImputer` with `n_neighbors=2` and `metric='nan_euclidean'`.
- Reconstruct the imputed DataFrame and reinsert the "ID" column.

KNN Imputation in Python: Step-by-Step

- Import necessary libraries: `pandas`, `numpy`, and `KNNImputer` from `sklearn`.
- Create a sample dataset with a missing value in the "Height (cm)" column.
- Drop the "ID" column to exclude it from distance calculations.
- Apply `KNNImputer` with `n_neighbors=2` and `metric='nan_euclidean'`.
- Reconstruct the imputed DataFrame and reinsert the "ID" column.
- Display the imputed dataset with the missing height value filled.

Reach PostNetwork Academy

Website

www.postnetwork.co

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

Facebook Page

www.facebook.com/postnetworkacademy

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

Facebook Page

www.facebook.com/postnetworkacademy

LinkedIn Page

www.linkedin.com/company/postnetworkacademy

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

Facebook Page

www.facebook.com/postnetworkacademy

LinkedIn Page

www.linkedin.com/company/postnetworkacademy

GitHub Repositories

www.github.com/postnetworkacademy

Thank You!