Artificial Intelligence

Iris Classification with K-Nearest Neighbor Algorithm

Fourier Survivor

Introducing our team

Fourier Survivor

Adhitya Primandhika
19/444033/TK/49229

Programmer 1,
Algoritma

Denny Dewanta
19/444045/TK/49241

Programmer 2,
PPT

Hafidz Arifudin
19/444052/TK/49248

Programmer 3,
Video

Introduction

Problem Exploration

Ditemukan bunga dengan karakteristik tertentu. Untuk mengidentifikasi satu bunga saja memerlukan waktu yang lama karena harus melihat berbagai macam ciri-ciri yang kemudian dicocokkan dengan karakteristik masing-masing spesies. Hal ini tentunya akan memakan waktu yang cukup lama untuk mengidentifikasi satu bunga, apalagi jika terdapat banyak bunga yang akan diidentifikasi.

Set Goals

Memprediksi suatu bunga termasuk ke dalam spesies apa berdasarkan karakteristik-karakteristik yang dimasukkan. Hasil keluaran berupa nama spesies.

Dataset Identification

150

Instances

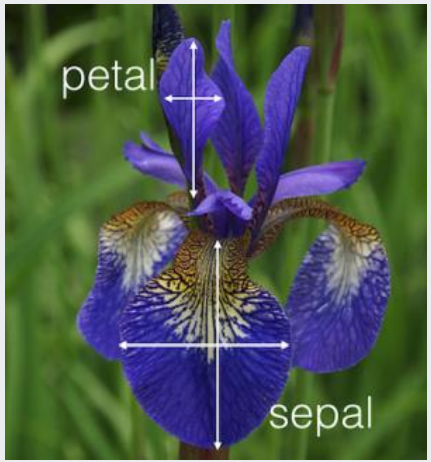
4

Features

1

Label:
Species

Dataset Identification: Features



01

Sepal Length (cm)

02

Sepal Width (cm)

03

Petal Length (cm)

04

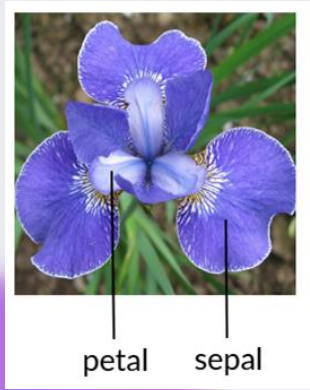
Petal Width (cm)

05

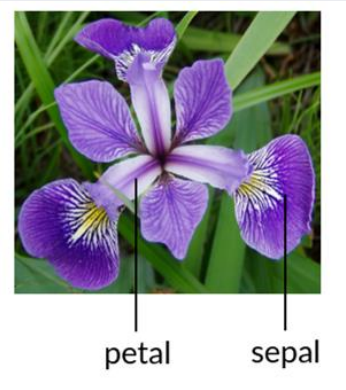
Species

Dataset Identification: Labels

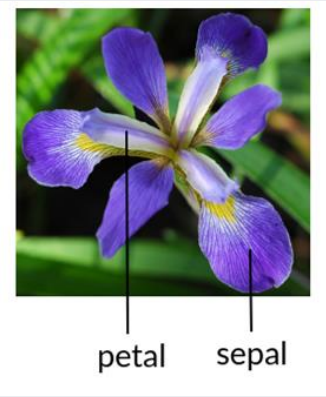
Iris **Setosa**



Iris **Versicolor**



Iris **Virginica**



https://miro.medium.com/max/1000/1*Hh53mOF4Xy4eORjLilKOwA.png

Method

K-Nearest Neighbor

Metode KNN adalah metode yang melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. KNN termasuk algoritma *supervised learning*. Apabila algoritma tersebut diberikan titik *query*, maka akan ditemukan sejumlah k objek atau titik latih yang paling dekat dengan titik query. Klasifikasi menggunakan *voting* terbanyak di antara klasifikasi dari k obyek.

Why KNN?

Mudah dipahami dan diimplementasikan

Hasil yang lebih akurat

Memiliki konsistensi yang kuat

Efektif apabila memiliki data *training sample* yang besar

KNN Algorithm

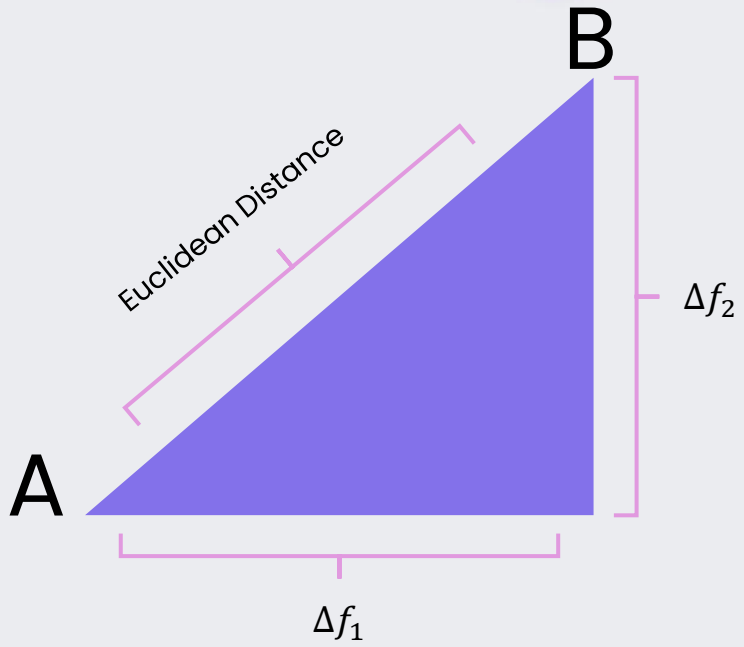
- Menentukan K yang akan digunakan
- Menghitung Euclidean distance
- Mengurutkan Euclidean distance secara ascending
- Didapatkan k rows dari *sorted array*
- Lalu, data dapat diprediksi

Classification Process: Compute Euclidean Distance

Suppose we have:

	f1	f2
A	2	3
B	1	5

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



Suppose we have:

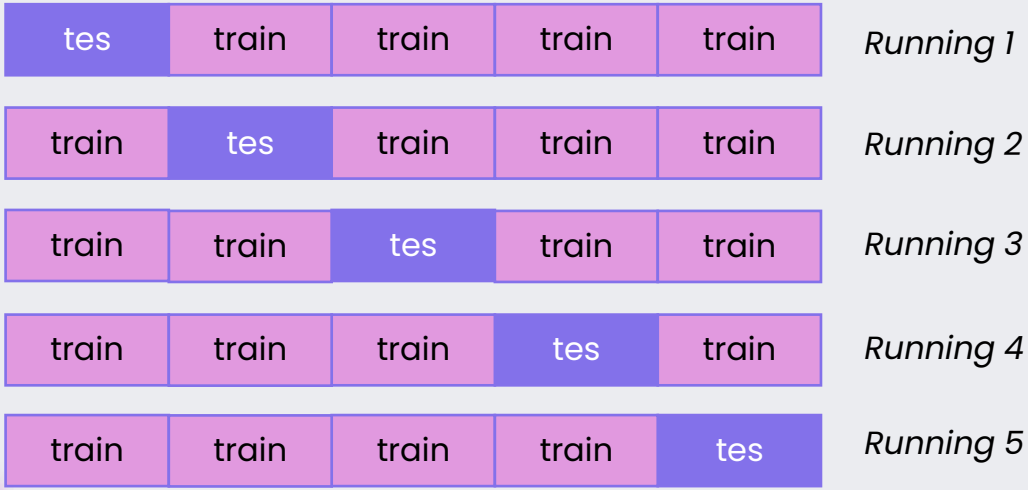
	f1	f2
A	2	3
B	1	5

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

$$d(A,B) = \sqrt{\sum_{i=1}^{2}(A_i - B_i)^2} = \sqrt{(2-1)^2 + (3-5)^2} = \sqrt{1+4} = \sqrt{5}$$

Cross Validation

Pada tugas ini digunakan k-*fold cross validation* dengan k = 5, sehingga proses *running* yang terjadi:



Implementation: Data Preparation

Import Libraries

```
# Import libraries
import pandas as pd
import numpy as np
import seaborn as sns
from math import sqrt
from collections import Counter
import scipy.spatial
import matplotlib.pyplot as plt
from pprint import pprint
```

Import Data

```
# Import data csv
df = pd.read_csv("/content/Iris.csv")
df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Check Null Values

```
# Check apakah ada nilai null atau tidak
df.isnull().sum()
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Plot Iris Data

```
# Membuat dataframe untuk plot data
setosa = df[df['Species']=='Iris-setosa']
versicolor = df[df['Species']=='Iris-versicolor']
virginica = df[df['Species']=='Iris-virginica']
```

Membuat *dataframe*

Plot Iris Data

```
# Plot data secara histogram
plt.figure()
fig,ax=plt.subplots(4,3,figsize=(17, 10))
setosa["SepalLengthCm"].plot(kind="hist", ax=ax[0][0],label="setosa",color ='salmon',fontsize=10)
versicolor["SepalLengthCm"].plot(kind="hist",ax=ax[0][1],label="versicolor",color='teal',fontsize=10)
virginica["SepalLengthCm"].plot(kind="hist",ax=ax[0][2],label="virginica",color='skyblue',fontsize=10)

setosa["SepalWidthCm"].plot(kind="hist", ax=ax[1][0],label="setosa",color ='salmon',fontsize=10)
versicolor["SepalWidthCm"].plot(kind="hist",ax=ax[1][1],label="versicolor",color='teal',fontsize=10)
virginica["SepalWidthCm"].plot(kind="hist",ax=ax[1][2],label="virginica",color='skyblue',fontsize=10)

setosa["PetalLengthCm"].plot(kind="hist", ax=ax[2][0],label="setosa",color ='salmon',fontsize=10)
versicolor["PetalLengthCm"].plot(kind="hist",ax=ax[2][1],label="versicolor",color='teal',fontsize=10)
virginica["PetalLengthCm"].plot(kind="hist",ax=ax[2][2],label="virginica",color='skyblue',fontsize=10)

setosa["PetalWidthCm"].plot(kind="hist", ax=ax[3][0],label="setosa",color ='salmon',fontsize=10)
versicolor["PetalWidthCm"].plot(kind="hist",ax=ax[3][1],label="versicolor",color='teal',fontsize=10)
virginica["PetalWidthCm"].plot(kind="hist",ax=ax[3][2],label="virginica",color='skyblue',fontsize=10)

plt.rcParams.update({'font.size': 10})
plt.tight_layout()
```

Plot histogram

Data Cleaning: Species to Number

```
# Mengganti spesies ke dalam bentuk angka
species = {'Iris-versicolor': 0,'Iris-virginica': 1, 'Iris-setosa': 2}
df.Species = [species[item] for item in df.Species]
df.head()
```

Split Data

```
# Membagi dataset menjadi training set dan test set. Digunakan training set 80% data
shuffle_df = df.sample(frac=1)
train_size = len(df)*0.8
train_df = shuffle_df[:int(train_size)]
test_df = shuffle_df[int(train_size):]
```

Implementation: KNN Class

KNN Class

```
[ ]  class KNN:
        def __init__(self, k):
            self.k = k

        def fit(self, X, y):
            self.x_train = X
            self.y_train = y

        def euclidean_distance(self, X1, X2):
            distance = 0.0
            for i in range(len(X1)-1):
                distance += (X1[i] - X2[i])**2
            return sqrt(distance)
```

KNN Class

```
def predict(self, x_test):
    predictions = []
    for i in range(len(x_test)):
        d = []
        votes = []
        for j in range(len(self.x_train)):
            dist = self.euclidean_distance(self.x_train[j] , x_test[i])
            d.append([dist, j])
        d.sort()
        d = d[0:self.k]
        for d, j in d:
            votes.append(self.y_train[j])
        ans = Counter(votes).most_common(1)[0][0]
        predictions.append(ans)

    return predictions
```

KNN Class

```
def score(self, x_test, y_test):
    predictions = self.predict(x_test)
    total = 0
    for i in range(len(y_test)):
      if predictions[i] == y_test[i]:
        total+=1
    return total / len(y_test)
```

Implementation: KNN

Reset Index

```
[ ]  # Melakukan reset index
     train_df.reset_index().drop(["index"],axis=1)
```

```
[ ]  # Melakukan reset index
     test_df.reset_index().drop(["index"],axis=1)
```

Data Preparation

```
# Menyiapkan x_train, y_train, x_test, dan y_test
x_train = train_df.drop(["Id","Species"], axis=1)
y_train = train_df["Species"]
x_test = test_df.drop(["Id","Species"], axis=1).copy()
y_test = test_df["Species"]
x_train.shape, y_train.shape, x_test.shape
```

Dataframe to Arrays

```
[ ]  # Mengubah dataframe menjadi bentuk array
     x_train =  np.array(x_train)
     y_train =  np.array(y_train)
     x_test =  np.array(x_test)
     y_test = np.array(y_test)
```

KNN Process

```
# Pemanggilan instance dari class KNN
model = KNN(8)
# Memanggil fungsi fit untuk menginisialisasi data yang akan di-train
model.fit(x_train, y_train)
# Memanggil fungsi predict untuk memprediksi nilai
prediction = model.predict(x_test)
prediction
```

Get Accuracy

```
[ ]  # Memanggil fungsi score untuk mendapatkan akurasi dari machine learning yang dibuat
     model.score(x_test,y_test)
```

Check Columns

```
# Melihat kolom apa saja yang ada pada dataframe df
df.columns
```

Create Dataframe

```
# Membuat dataframe result yang berisi features
result = pd.DataFrame(x_test,columns=["SepalLengthCm", "SepalWidthCm", "PetalLengthCm", "PetalWidthCm"])
# Membuat dataframe species yang berisi hasil prediksi
species = pd.DataFrame(prediction, columns=["Species"])
# Melihat bentuk dari dataframe
result.shape, species.shape
```

Add Predictions

```
[ ]  # Menambah kolom Species dengan data pada dataframe species
     result["Species"] = species
```

View The Data

```
# Melihat 5 data teratas pada dataframe result
result.head()
```

Mapping Values

```
[ ]  # Melakukan mapping value dari kolom spesies dari angka menjadi dalam bentuk kata jenis spesies
     species_mapping = {0 : 'Iris-versicolor', 1 :'Iris-virginica', 2: 'Iris-setosa'}
     result["Species"] = result["Species"].map(species_mapping).astype(str)
     result.head()
```

Plot Data

```
[ ]  # Plot untuk melihat gambaran sebaran tiap spesies
     sns.pairplot(data=result, hue="Species")
```

Cross Validation

Fold The Dataframe

```
[72] fold = [shuffle_df[0:29],shuffle_df[30:59],shuffle_df[60:89],shuffle_df[90:119],shuffle_df[120:149]]
     train = []
     test = []
     cross_val={'train': train, 'test': test}
     for i, j in enumerate(fold):
         train.append(fold[:i] + fold[i+1:])
         test.append(j)
     pprint(cross_val)
```

```
train_list = cross_val["train"]
train_list
```

```
[74] test_list = cross_val["test"]
     test_list
```

Run and Combine

```
[212] predictions = []
      scores = []
      # iterasi untuk me-running
      for i in range(5):
        temp = []
        if i!=0:
          for j in range(i):
            temp = temp + train_list[j]
        for k in range(5-(i+1)):
          temp = temp + train_list[k+i+1]
        train_temp = temp
        test_temp = test_list[i]
        train_temp = pd.concat(train_temp)
```

Implement the KNN

```
x_train = np.array(train_temp.drop(["Id","Species"], axis=1))
y_train = np.array(train_temp["Species"])
x_test = np.array(test_temp.drop(["Id","Species"], axis=1).copy())
y_test = np.array(test_temp["Species"])
model = KNN(8)
model.fit(x_train, y_train)
prediction = model.predict(x_test)
score = model.score(x_test,y_test)
predictions.append(prediction)
scores.append(score)
```

Get Accuracies

```
[215] # Menghitung akurasi rata-rata
      total = 0
      for score in scores:
        total+=score

      average_accuracy = total/len(scores)
      print(str(average_accuracy*100)+'%')
```

Evaluation

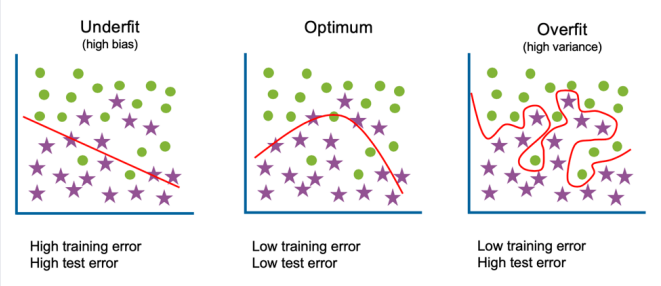
Evaluation: Accuracy

```
[214] scores

      [1.0, 1.0, 1.0, 1.0, 1.0]

[215] # Menghitung akurasi rata-rata
      total = 0
      for score in scores:
        total+=score

      average_accuracy = total/len(scores)
      print(str(average_accuracy*100)+'%')

      100.0%
```

100% accuracy? Is it good?

How overfitting can be avoided?

- Early stopping
- Train with more data
- Data augmentation
- Feature selection
- Regularization
- Ensemble methods



Gambar dari ibm.com/cloud/learn/overfitting

Summary

References

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

IBM Cloud Education. (2021, March 3). *What is Overfitting?* Ibm.com. https://www.ibm.com/cloud/learn/overfitting [Accessed 9 June 2021].

MasChoi (2018). *K-Nearest Neighbors Menggunakan Python - BOSBOUW - Medium*. [online] Medium. Available at: https://medium.com/bosbouw/k-nearest-neighbors-menggunakan-python-bd3652ba1e70 [Accessed 7 June. 2021].

Fourier Survivor

Thank You!