**Pertemuan 5 dan 6 – Algoritma Klasifikasi Data**

**Tujuan pembelajaran**

* Mahasiswa mampu memahami beberapa algoritma klasifikasi seperti decision tree, naïve bayes, k-nearest neighbor, support vector machine dan logistik regresi.
* Mahasiswa mampu memahami perbedaan antara algoritma klasifikasi decision tree, naïve bayes, k-nearest neighbor, support vector machine dan logistik regresi.

**Studi kasus: Klasifikasi Bunga Iris Menggunakan Metode Supervised Learning**

|  |
| --- |
| # lib data manipulations  import numpy as np  import pandas as pd    # lib data visualizaton  import seaborn as sns  import matplotlib.pyplot as plt    # lib data preprocessing  from sklearn.preprocessing import MinMaxScaler  from sklearn.model\_selection import train\_test\_split    # lib supervised learning  from sklearn.neighbors import KNeighborsClassifier  from sklearn.tree import DecisionTreeClassifier  from sklearn.naive\_bayes import GaussianNB  from sklearn.linear\_model import LogisticRegression  from sklearn.svm import SVC    # library evaluation model  from sklearn.metrics import classification\_report, confusion\_matrix  from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score |

1. **Akuisisi Data**

|  |
| --- |
| # load dataset  dataset = pd.read\_csv("../dataset/iris.csv")  # show metadata  print(np.round(dataset.describe(),2)) |
| sepal\_length sepal\_width petal\_length petal\_width  count 150.00 150.00 150.00 150.00  mean 5.84 3.05 3.76 1.20  std 0.83 0.43 1.76 0.76  min 4.30 2.00 1.00 0.10  25% 5.10 2.80 1.60 0.30  50% 5.80 3.00 4.35 1.30  75% 6.40 3.30 5.10 1.80  max 7.90 4.40 6.90 2.50 |

1. **Visualisasi Data**

|  |
| --- |
| # define boxplot  fig, ax = plt.subplots(figsize=(8,4))  ax.boxplot(  dataset[["sepal\_length", "sepal\_width", "petal\_length", "petal\_width"]],  labels=["sepal\_length", "sepal\_width", "petal\_length", "petal\_width"],  patch\_artist=True, widths=(0.75, 0.75, 0.75, 0.75)  )    # set labels  ax.set\_title('')  ax.set\_xlabel('')  ax.set\_ylabel('')  ax.grid(True)    # show boxplot  plt.tight\_layout()  plt.show() |
|  |
| Gambar x. Output program |

|  |
| --- |
| # create barplot  fig, ax = plt.subplots(figsize=(8,4))  sns.countplot(dataset, x="species", hue="species")    # set labels  ax.set\_title("", fontsize=14)  ax.set\_xlabel("", fontsize=12)  ax.set\_ylabel("", fontsize=12)  ax.grid(True)    # show plot  plt.tight\_layout()  plt.show() |
|  |
| Gambar x. Output program |

|  |
| --- |
| # function scatter plot  def scatter(data, x, y):      # create scatter plots    fig, ax = plt.subplots(figsize = (8,4))    sns.scatterplot(data=data, x=x, y=y, hue="species")      # set labels    ax.set\_title("", fontsize=14)    ax.set\_xlabel("", fontsize=12)    ax.set\_ylabel("", fontsize=12)    ax.legend(loc='upper left')    ax.grid(True)  plt.tight\_layout()      # show plots    plt.tight\_layout()    plt.show() |

|  |
| --- |
| # call function scatter  scatter(dataset, "petal\_length", "sepal\_length") |
|  |
| Gambar x. Output program |

|  |
| --- |
| # call function scatter  scatter(dataset, "petal\_length", "petal\_width") |
|  |
| Gambar x. Output program |

|  |
| --- |
| # create heatmap corr  fig, ax = plt.subplots(figsize=(8,4))  sns.heatmap(    dataset.corr(numeric\_only=True), vmin=-1, vmax=1,    cmap="viridis", annot=True, fmt=".3f", linewidths=1)    # set labels  ax.set\_title("", fontsize=14)  ax.set\_xlabel("", fontsize=12)  ax.set\_ylabel("", fontsize=12)    # show plot  plt.show() |
|  |
| Gambar x. Output program |

|  |
| --- |
| # create kdeplot  fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12,4))  sns.kdeplot(data=dataset, x="sepal\_width", hue="species", fill=True, ax=ax[0])  ax[0].set\_title("", fontsize=14)  ax[0].set\_xlabel("", fontsize=12)  ax[0].set\_ylabel("", fontsize=12) ax[0].grid(True)    # create kdeplot  sns.kdeplot(data=dataset, x="petal\_width", y="sepal\_width", hue="species", fill=True, ax=ax[1])  ax[1].set\_title("", fontsize=14)  ax[1].set\_xlabel("", fontsize=12)  ax[1].set\_ylabel("", fontsize=12)  ax[1].grid(True)    # show plots  plt.tight\_layout()  plt.show() |
|  |
| Gambar x. Output program |

1. **Praproses Data**

|  |
| --- |
| # Set features and Labels  x = dataset[["sepal\_length","sepal\_width","petal\_length","petal\_width"]].values  y = dataset["species"].values |

|  |
| --- |
| # normalize features  scaler = MinMaxScaler(feature\_range=(0, 1))  scaled = scaler.fit\_transform(x) |

|  |
| --- |
| # split validation  trainX, testX, trainY, testY = train\_test\_split(    scaled, y, train\_size=0.7, test\_size=0.3, random\_state=7, shuffle=True  ) |

|  |
| --- |
| # show dimension of data train  print(trainX.shape, trainY.shape) |
| (105, 4) (105,) |

|  |
| --- |
| # show dimension of data test  print(testX.shape, testY.shape) |
| (45, 4) (45,) |

1. **Decision Tree dengan C45**

|  |
| --- |
| # supervised learning algorithm  result\_C45 = DecisionTreeClassifier(criterion="gini", random\_state=None).fit(trainX, trainY).predict(testX) |

|  |
| --- |
| # evaluation models - confusion\_matrix  conf\_C45 = confusion\_matrix(testY, result\_C45)  conf\_C45 |
| array([[12, 0, 0],  [ 0, 14, 2],  [ 0, 2, 15]], dtype=int64) |

|  |
| --- |
| # evaluation models - classification\_report  print(classification\_report(y\_true=testY, y\_pred=result\_C45)) |
| precision recall f1-score support  setosa 1.00 1.00 1.00 12  versicolor 0.88 0.88 0.88 16  virginica 0.88 0.88 0.88 17  accuracy 0.91 45  macro avg 0.92 0.92 0.92 45  weighted avg 0.91 0.91 0.91 45 |

1. **Naive Bayes - Gaussian**

|  |
| --- |
| # supervised learning algorithm  result\_gnb = GaussianNB().fit(trainX, trainY).predict(testX) |

|  |
| --- |
| # evaluation models - confusion\_matrix  conf\_gnb = confusion\_matrix(testY, result\_gnb)  conf\_gnb |
| array([[12, 0, 0],  [ 0, 13, 3],  [ 0, 2, 15]], dtype=int64) |

|  |
| --- |
| # evaluation models - classification\_report  print(classification\_report(y\_true=testY, y\_pred=result\_gnb)) |
| precision recall f1-score support  setosa 1.00 1.00 1.00 12  versicolor 0.87 0.81 0.84 16  virginica 0.83 0.88 0.86 17  accuracy 0.89 45  macro avg 0.90 0.90 0.90 45  weighted avg 0.89 0.89 0.89 45 |

1. **K Nearest-Neighbor**

|  |
| --- |
| # supervised learning algorithm  result\_knn = KNeighborsClassifier(n\_neighbors=3).fit(trainX, trainY).predict(testX) |

|  |
| --- |
| # evaluation models - confusion\_matrix  conf\_knn = confusion\_matrix(testY, result\_knn)  conf\_knn |
| array([[12, 0, 0],  [ 0, 16, 0],  [ 0, 2, 15]], dtype=int64) |

|  |
| --- |
| # evaluation models - classification\_report  print(classification\_report(y\_true=testY, y\_pred=result\_knn)) |
| precision recall f1-score support  setosa 1.00 1.00 1.00 12  versicolor 0.89 1.00 0.94 16  virginica 1.00 0.88 0.94 17  accuracy 0.96 45  macro avg 0.96 0.96 0.96 45  weighted avg 0.96 0.96 0.96 45 |

1. **Logistic Regression**

|  |
| --- |
| # supervised learning algorithm  result\_lr = LogisticRegression(max\_iter=1000).fit(trainX, trainY).predict(testX) |

|  |
| --- |
| # evaluation models - confusion\_matrix  conf\_lr = confusion\_matrix(testY, result\_lr)  conf\_lr |
| array([[12, 0, 0],  [ 0, 11, 5],  [ 0, 2, 15]], dtype=int64) |

|  |
| --- |
| # evaluation models - classification\_report  print(classification\_report(y\_true=testY, y\_pred=result\_lr) |
| precision recall f1-score support  setosa 1.00 1.00 1.00 12  versicolor 0.85 0.69 0.76 16  virginica 0.75 0.88 0.81 17  accuracy 0.84 45  macro avg 0.87 0.86 0.86 45  weighted avg 0.85 0.84 0.84 45 |

1. **Support Vector Classifier**

|  |
| --- |
| # supervised learning algorithm  result\_svc = SVC(kernel='rbf').fit(trainX, trainY).predict(testX) |

|  |
| --- |
| # evaluation models - confusion\_matrix  conf\_svc = confusion\_matrix(testY, result\_svc)  conf\_svc |
| array([[12, 0, 0],  [ 0, 14, 2],  [ 0, 1, 16]], dtype=int64) |

|  |
| --- |
| # evaluation models - classification\_report  print(classification\_report(y\_true=testY, y\_pred=result\_svc)) |
| precision recall f1-score support  setosa 1.00 1.00 1.00 12  versicolor 0.93 0.88 0.90 16  virginica 0.89 0.94 0.91 17  accuracy 0.93 45  macro avg 0.94 0.94 0.94 45  weighted avg 0.93 0.93 0.93 45 |

**Selesai, Selamat Mencoba :3**