KECERDASAN BUATAN

“PRAKTIKUM 7 – Jaringan Syaraf Tiruan”

Disusun oleh :

Mario Rizky Saputra_G.231.20.0142

PROGRAM STUDI S1-TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI

UNIVERSITAS SEMARANG

2023

```
#!pip install keras
#Remove '#' in above line if not installed keras
import keras
import numpy as np
from sklearn.datasets import load_iris
dataset=load_iris()
##How data looks like?
print(dataset)
```

- `#!pip install keras`: Baris ini digunakan untuk menginstal library keras jika belum terinstal. Tanda `#!` menunjukkan perintah shell yang dijalankan dalam lingkungan Python.
- 2. `import keras`: Baris ini mengimpor library keras yang sudah diinstal sebelumnya. Keras adalah library digunakan untuk membangun dan melatih model deep learning.
- `import numpy as np`: Baris ini mengimpor library numpy dengan alias np. Numpy adalah library yang digunakan untuk melakukan operasi numerik efisien pada array multidimensi.
- `from sklearn.datasets import load_iris`: Baris ini mengimpor fungsi `load_iris` dari library sklearn.datasets. Fungsi ini digunakan untuk memuat dataset Iris, yang merupakan dataset yang umum digunakan dalam pembelajaran mesin.
- `dataset=load_iris()`: Baris ini memuat dataset Iris menggunakan fungsi `load_iris()` dan menyimpannya ke dalam variabel `dataset`.
- `print(dataset)`: Baris ini mencetak isi dari variabel `dataset`, yang berisi informasi tentang dataset Iris, termasuk atribut-atributnya dan labelnya.

```
##Step : 1 : Know the data
print(type(dataset))
print(len(dataset))
#Dataset is in bunch format. Bunch is a dictionary like object.
print(dataset.keys())
#We try to find out what values are stored in these attributes
print(dataset['data'][0:5])
print(dataset['target'][0:5])
print(dataset['target_names'][0:5])
print(dataset['DESCR'])
print(dataset['feature_names'])
print(dataset['filename'])
```

```
#Next we define our x and y;  y is the dependent on x
x=dataset['data']
print(len(x)) #To check length of dataset
y=dataset['target']
print(len(y)) #To check length of targets
print(x[0])
print(y[0])
```

- `print(type(dataset))` mencetak jenis data dari variabel `dataset`. Ini digunakan untuk mengetahui jenis atau tipe data yang digunakan.
- `print(len(dataset))` mencetak panjang atau jumlah elemen dalam `dataset`. Ini memberikan informasi tentang ukuran dataset.
- `print(dataset.keys())` mencetak kunci atau atribut yang ada dalam `dataset`. Pada script ini, `dataset` adalah objek tipe kamus (dictionary-like object), sehingga `keys()` digunakan untuk menampilkan semua kunci yang ada dalam kamus tersebut.
- `print(dataset['data'][0:5])` mencetak nilai dari atribut `'data'` dalam `dataset`. Di sini, mencetak 5 elemen pertama dari atribut `'data'`. Ini memberikan gambaran tentang struktur dan isi data dalam atribut `'data'`.
- `print(dataset['target'][0:5])` mencetak nilai dari atribut `'target'` dalam `dataset`. Seperti langkah sebelumnya, mencetak 5 elemen pertama dari atribut `'target'`. Ini memberikan gambaran tentang struktur dan isi target yang sesuai dengan data dalam atribut `'data'`.
- `print(dataset['target_names'][0:5])` mencetak nilai dari atribut `'target_names'` dalam `dataset`. Atribut ini berisi nama-nama yang sesuai dengan nilai target. Dalam script ini mencetak 5 nama pertama dari atribut `'target_names'`.
- `print(dataset['DESCR'])` mencetak penjelasan tentang dataset. Atribut `'DESCR'` berisi informasi rinci tentang dataset yang digunakan.
- `print(dataset['feature_names'])` mencetak nilai dari atribut `'feature_names'` dalam `dataset`. Atribut ini berisi nama-nama fitur atau atribut yang digunakan dalam dataset.
- `print(dataset['filename'])` mencetak nilai dari atribut `'filename'` dalam `dataset`. Atribut ini berisi nama file dari dataset yang digunakan.

- `x = dataset['data']` menginisialisasi variabel `x` dengan nilai atribut `'data'` dalam `dataset`. Variabel `x` akan digunakan sebagai fitur atau atribut yang menjadi input dalam model.
- `print(len(x))` mencetak panjang atau jumlah elemen dalam variabel `x`. Ini memberikan informasi tentang ukuran dataset dalam variabel `x`.
- `y = dataset['target']` menginisialisasi variabel `y` dengan nilai atribut `'target'` dalam `dataset`. Variabel `y` akan digunakan sebagai output yang ingin diprediksi dalam model.
- `print(len(y))` mencetak panjang atau jumlah elemen dalam variabel `y`. Ini memberikan informasi tentang ukuran target dalam variabel `y`.
- `print(x[0])` mencetak nilai pertama dalam variabel `x`. Ini memberi gambaran struktur data dalam variabel `x`.
- `print(y[0])` mencetak nilai pertama dalam variabel `y`. Ini memberi gambaran format struktur target dalam variabel `y`.

```
##Step : 2 : Convert y into one hot encoded vector
#One hot encoding is used because Y has labels (0,1,2)- which is a
categorical categorical data with no ordinal relationships
from keras.utils import to_categorical
Ny=len(np.unique(y))
print(Ny)
Y=to_categorical(y,num_classes=Ny)
print(Y[0:5])
```

- `from keras.utils import to_categorical`: Ini mengimpor fungsi `to_categorical` dari pustaka Keras. Fungsi ini digunakan untuk melakukan one-hot encoding.
- `Ny=len(np.unique(y))`: Baris ini menghitung jumlah kategori dalam vektor target `y`. `np.unique(y)` mengembalikan nilai dalam `y`, dan `len(np.unique(y))` menghitung jumlah elemen. Hasilnya disimpan dalam variabel `Ny`.
- `print(Ny)`: Baris ini mencetak jumlah kategori (`Ny`) ke konsol.
- `Y=to_categorical(y,num_classes=Ny)`: Fungsi `to_categorical` digunakan untuk mengubah vektor target `y` menjadi representasi one-hot encoding. Parameter `y`

adalah vektor target yang akan diubah, dan `num_classes=Ny` menentukan jumlah kategori yang diinginkan. Hasilnya disimpan dalam variabel `Y`.

- `print(Y[0:5])`: Baris ini mencetak lima baris pertama dari representasi one-hot encoding (`Y`) ke konsol. Representasi one-hot encoding adalah matriks setiap baris menggambarkan kategori target yang diubah menjadi vektor dengan nilai 1 pada indeks kategori yang sesuai, dan 0 pada indeks kategori lainnya

```
##Step : 3 : Now we split the data into two parts - one for training
another for testing
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,Y,test_size=0.10,shuffle=
True)
print(x_train[0:5])
print(x_test[0:5])
print(y_train[0:5])
print(y_test[0:5])
```

- `from sklearn.model_selection import train_test_split`: Baris ini mengimpor fungsi `train_test_split` dari modul `model_selection` dalam pustaka `sklearn`. Ini digunakan untuk membagi dataset menjadi subset pelatihan dan pengujian.
- `x_train, x_test, y_train, y_test = train_test_split(x, Y, test_size=0.10, shuffle=True)`: Baris ini melakukan pemanggilan fungsi `train_test_split` untuk membagi dua set data, `x` dan `Y`, menjadi empat subset yang berbeda: `x_train`, `x_test`, `y_train`, dan `y_test`.
    - `x` adalah input features atau atribut-atribut dari dataset.
    - `Y` adalah target atau label yang sesuai dengan setiap input.
    - `test_size=0.10` menentukan bahwa ukuran subset pengujian adalah 10% dari keseluruhan dataset. Artinya, 90% data akan digunakan sebagai subset pelatihan.
    - `shuffle=True` mengacaukan urutan data sebelum membaginya menjadi subset. Ini berguna untuk mencegah bias yang mungkin timbul jika data terurut.
- `print(x_train[0:5])`: Baris ini mencetak lima baris pertama dari subset pelatihan `x_train`. Untuk memeriksa apakah pembagian data.

- `print(x_test[0:5])`: Baris ini mencetak lima baris pertama dari subset pengujian `x_test`. Untuk memeriksa apakah pembagian data.
- `print(y_train[0:5])`: Baris ini mencetak lima baris pertama dari subset pelatihan `y_train`. Untuk memeriksa apakah pembagian data.
- `print(y_test[0:5])`: Baris ini mencetak lima baris pertama dari subset pengujian `y_test`. Untuk memeriksa apakah pembagian data.

```
##Step : 4 : Then we normalize the data
from sklearn.preprocessing import StandardScaler
##Normalization is done so that the difference between highest and lowest
data point is not too large
import numpy as np
scaler=StandardScaler()
##To find mean and std dev
scaler.fit(x_train)
## Please Note - We are using mean and std dev of training data for
testing data too. It is done to ensure that no information of test data is
leaked to the model.
##Converting data into form where mean of data is 0 and std dev is 1
X_train=scaler.transform(x_train)
X_test=scaler.transform(x_test)
print(np.amax(X_train,axis=0))
print(np.amin(X_train,axis=0))
```

- `from sklearn.preprocessing import StandardScaler` - Mengimpor kelas `StandardScaler` dari pustaka `sklearn.preprocessing`. Kelas ini digunakan untuk melakukan standarisasi data.
- `import numpy as np` - Mengimpor pustaka NumPy sebagai `np`. NumPy adalah pustaka yang digunakan untuk operasi array dan komputasi numerik.
- `scaler = StandardScaler()` - Membuat objek `StandardScaler` yang akan digunakan untuk melakukan standarisasi data.
- `scaler.fit(x_train)` - Menggunakan data pelatihan (`x_train`) untuk menghitung mean dan standar deviasi yang akan digunakan. Dalam langkah ini, `x_train` sebagai matriks fitur yang disesuaikan dengan skala.

- `X_train = scaler.transform(x_train)` - Melakukan standarisasi pada data pelatihan (`x_train`) dengan menggunakan skala yang telah dihitung sebelumnya. Hasil standarisasi disimpan dalam `X_train`.
- `X_test = scaler.transform(x_test)` - Melakukan standarisasi pada data uji (`x_test`) dengan menggunakan skala yang sama yang dihitung dari data pelatihan. Hasil standarisasi disimpan dalam `X_test`
- `print(np.amax(X_train, axis=0))` - Mencetak nilai maksimum dari setiap kolom matriks `X_train`. Fungsi `np.amax()` digunakan untuk menghitung nilai maksimum di sepanjang sumbu yang ditentukan.
- `print(np.amin(X_train, axis=0))` - Mencetak nilai minimum dari setiap kolom dalam matriks `X_train`. Fungsi `np.amin()` digunakan untuk menghitung nilai minimum di sepanjang sumbu yang ditentukan.

```
##Step : 5 : Now finally we build the model using keras
!pip install tensorflow
import keras
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(20, input_dim=X_train.shape[1], activation='relu'))
##Dropout is used to avoid overfitting
keras.layers.Dropout(0.2)
model.add(Dense(20, activation='relu'))
keras.layers.Dropout(0.2)
model.add(Dense(20, activation='relu'))
keras.layers.Dropout(0.2)
##For classification problems, we usually use softmax as activation
function in final layer
model.add(Dense(3, activation='softmax'))
##Compiling the model
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
```

- `!pip install tensorflow`: Untuk menginstal library TensorFlow jika belum terinstal.

- `import keras`: Baris ini mengimpor modul `keras` yang merupakan antarmuka tingkat tinggi untuk membangun dan melatih model deep learning. Keras adalah bagian dari TensorFlow.
- `from keras.models import Sequential`: Ini mengimpor kelas `Sequential` dari modul `models` di Keras. Kelas `Sequential` digunakan untuk membuat model neural network secara berurutan.
- `from keras.layers import Dense`: Ini mengimpor kelas `Dense` dari modul `layers` di Keras. Kelas `Dense` digunakan untuk menambahkan lapisan-lapisan densely connected (sepenuhnya terhubung) ke dalam model neural network.
- `model = Sequential()`: Membuat objek model dari kelas `Sequential`. Objek ini akan digunakan untuk menambahkan lapisan-lapisan ke dalam model secara berurutan.
- `model.add(Dense(20,input_dim=X_train.shape[1],activation='relu'))`: Menambahkan lapisan pertama ke dalam model.
- `model.add(Dense(20, activation='relu'))`: Menambahkan lapisan Dense kedua ke dalam model.
- `model.add(Dense(3, activation='softmax'))`: Menambahkan lapisan Dense terakhir ke dalam model.
- `model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accurac'])`: Mengompilasi model dengan menentukan fungsi loss, optimizer, dan metrik evaluasi.

```
##Step : 6 : Know the model
model.summary()
```

- Kode **model.summary()** digunakan untuk mencetak ringkasan (summary) dari arsitektur model yang telah didefinisikan.

```
##Step : 7 : Train the model
##Batch size,epochs and validation split are all hyper parameters
model.fit(X_train, y_train, epochs=195, batch_size=80,
validation_split=0.1)
```

- **X_train** adalah data pelatihan yang digunakan untuk melatih model.

- **y_train** adalah label atau target yang sesuai dengan data pelatihan.
- **epochs=195** menentukan jumlah iterasi atau kali melalui seluruh dataset pelatihan yang akan dilakukan saat melatih model.
- **batch_size=80** menentukan jumlah sampel data yang akan digunakan pada setiap iterasi. Model akan memproses **batch_size** sampel pada setiap langkah dalam satu iterasi sebelum memperbarui bobot.
- **validation_split=0.1** menunjukkan proporsi data pelatihan yang akan digunakan untuk validasi model. Dalam hal ini, 10% data pelatihan akan digunakan sebagai data validasi.

```
##Step : 8 : Get the accuracy of model on testing data
testing=model.evaluate(X_test, y_test)
print("\n%s: %.2f%%" % (model.metrics_names[1]+'uracy of Model on testing data', testing[1]*100))
```

- `testing=model.evaluate(X_test, y_test)`: Kode ini menghasilkan evaluasi model menggunakan metode `evaluate` pada data pengujian (`X_test`) dan label yang sesuai (`y_test`). Hasil evaluasi ini akan disimpan dalam variabel `testing`.
- `print("\n%s: %.2f%%" % (model.metrics_names[1]+'uracy of Model on testing data', testing[1]*100))`: Kode ini mencetak akurasi model pada data pengujian. `%s` dan `%f` adalah placeholder yang akan diisi dengan nilai yang sesuai. `%s` akan diisi dengan string yang menggabungkan nama metrik dari model dengan teks tambahan "Accuracy of Model on testing data". `%f` akan diisi dengan nilai akurasi dari evaluasi model (`testing[1]`). Notasi `%.2f` menunjukkan bahwa angka desimal akan dicetak dengan dua digit di belakang koma. Operator `%` digunakan untuk memformat string dengan nilai yang diinginkan.

```
##Step : 9 : Evaluate our model
from sklearn.metrics import classification_report, confusion_matrix
predictions = np.argmax(model.predict(X_test), axis=1)
Y_test = np.argmax(y_test,axis=1)
#To get confusion matrix
```

```
print(confusion_matrix(Y_test,predictions))
#To get values of all evaluation metrics
print(classification_report(Y_test,predictions))
```

- `from sklearn.metrics import classification_report, confusion_matrix`: Kode ini mengimpor fungsi `classification_report` dan `confusion_matrix` dari modul `metrics` di pustaka Scikit-learn. Fungsi-fungsi ini digunakan untuk mengevaluasi hasil prediksi model klasifikasi.
- `predictions = np.argmax(model.predict(X_test), axis=1)`: Untuk melakukan prediksi menggunakan model `model` terhadap data uji `X_test`. Fungsi `argmax` digunakan untuk mengambil indeks dengan nilai maksimum dari setiap baris prediksi, sehingga menghasilkan array `predictions` yang berisi label prediksi untuk setiap sampel data uji.
- `Y_test = np.argmax(y_test, axis=1)`: Kode ini mengubah array `y_test` (label sebenarnya) menjadi bentuk yang sama seperti array `predictions` dengan menggunakan fungsi `argmax`. Tujuannya adalah untuk membandingkan label prediksi dengan label sebenarnya saat melakukan evaluasi.
- `print(confusion_matrix(Y_test,predictions))`: Kode ini mencetak matriks (confusion matrix) yang menggambarkan perbandingan antara label prediksi dan label sebenarnya.
- `print(classification_report(Y_test,predictions))`: Kode ini mencetak laporan klasifikasi yang menyediakan metrik evaluasi seperti presisi, recall, f1-score, dan dukungan untuk setiap kelas. Laporan klasifikasi ini memberikan gambaran lebih lengkap tentang kinerja model klasifikasi.