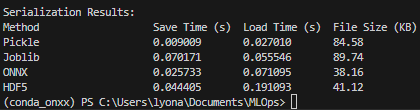
1. **Kode**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122 | import pandas as pd  import onnx  import pickle  import joblib  import tensorflow as tf  import time  import os  from sklearn.datasets import load\_iris  from sklearn.ensemble import RandomForestClassifier  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import accuracy\_score  from skl2onnx import convert\_sklearn  from skl2onnx.common.data\_types import FloatTensorType  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Dense  from sklearn.preprocessing import OneHotEncoder, StandardScaler  iris = load\_iris()  X = pd.DataFrame(iris.data, columns=iris.feature\_names)  y = pd.DataFrame(iris.target, columns=["target"])  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  rf\_model = RandomForestClassifier(n\_estimators=50, random\_state=42)  rf\_model.fit(X\_train, y\_train.values.ravel())  y\_pred = rf\_model.predict(X\_test)  accuracy = accuracy\_score(y\_test, y\_pred)  print(f"Random Forest Model Accuracy: {accuracy}")  scaler = StandardScaler()  X\_scaled = scaler.fit\_transform(X)  encoder = OneHotEncoder(sparse=False)  y\_encoded = encoder.fit\_transform(y)  X\_train\_tf, X\_test\_tf, y\_train\_tf, y\_test\_tf = train\_test\_split(X\_scaled, y\_encoded, test\_size=0.3, random\_state=42)  tf\_model = Sequential([  Dense(32, activation='relu', input\_shape=(X\_train\_tf.shape[1],)),  Dense(16, activation='relu'),  Dense(y\_train\_tf.shape[1], activation='softmax')  ])  tf\_model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])  tf\_model.fit(X\_train\_tf, y\_train\_tf, epochs=20, batch\_size=8, verbose=0)  y\_pred\_tf = tf\_model.predict(X\_test\_tf)  y\_pred\_tf\_classes = tf.argmax(y\_pred\_tf, axis=1).numpy()  y\_test\_tf\_classes = tf.argmax(y\_test\_tf, axis=1).numpy()  tf\_accuracy = accuracy\_score(y\_test\_tf\_classes, y\_pred\_tf\_classes)  print(f"TensorFlow Model Accuracy: {tf\_accuracy}")  serialization\_results = []  pickle\_file = "random\_forest\_pickle.pkl"  start\_save = time.time()  with open(pickle\_file, "wb") as file:  pickle.dump(rf\_model, file)  end\_save = time.time()  pickle\_save\_time = end\_save - start\_save  start\_load = time.time()  with open(pickle\_file, "rb") as file:  pickle\_model = pickle.load(file)  end\_load = time.time()  pickle\_load\_time = end\_load - start\_load  pickle\_file\_size = os.path.getsize(pickle\_file) / 1024  serialization\_results.append(("Pickle", pickle\_save\_time, pickle\_load\_time, pickle\_file\_size))  joblib\_file = "random\_forest\_joblib.pkl"  start\_save = time.time()  joblib.dump(rf\_model, joblib\_file)  end\_save = time.time()  joblib\_save\_time = end\_save - start\_save  start\_load = time.time()  joblib\_model = joblib.load(joblib\_file)  end\_load = time.time()  joblib\_load\_time = end\_load - start\_load  joblib\_file\_size = os.path.getsize(joblib\_file) / 1024  serialization\_results.append(("Joblib", joblib\_save\_time, joblib\_load\_time, joblib\_file\_size))  onnx\_file = "random\_forest.onnx"  initial\_type = [('float\_input', FloatTensorType([None, X\_train.shape[1]]))]  onnx\_model = convert\_sklearn(rf\_model, initial\_types=initial\_type)  start\_save = time.time()  with open(onnx\_file, "wb") as file:  file.write(onnx\_model.SerializeToString())  end\_save = time.time()  onnx\_save\_time = end\_save - start\_save  start\_load = time.time()  onnx\_loaded\_model = onnx.load(onnx\_file)  end\_load = time.time()  onnx\_load\_time = end\_load - start\_load  onnx\_file\_size = os.path.getsize(onnx\_file) / 1024  serialization\_results.append(("ONNX", onnx\_save\_time, onnx\_load\_time, onnx\_file\_size))  h5\_file = "tensorflow\_model.h5"  start\_save = time.time()  tf\_model.save(h5\_file, save\_format='h5')  end\_save = time.time()  h5\_save\_time = end\_save - start\_save  start\_load = time.time()  loaded\_h5\_model = tf.keras.models.load\_model(h5\_file)  end\_load = time.time()  h5\_load\_time = end\_load - start\_load  h5\_file\_size = os.path.getsize(h5\_file) / 1024  serialization\_results.append(("HDF5", h5\_save\_time, h5\_load\_time, h5\_file\_size))  print("\nSerialization Results:")  print(f"{'Method':<25}{'Save Time (s)':<15}{'Load Time (s)':<15}{'File Size (KB)':<15}")  for method, save\_time, load\_time, file\_size in serialization\_results:  print(f"{method:<25}{save\_time:<15.6f}{load\_time:<15.6f}{file\_size:<15.2f}") |

1. **Screenshot dan Penjelasan**



Dilakukan 4 metode serialisasi, yaitu pickle, joblib, ONNX, dan model tensorflow yang disimpan dalam format HDF5 (.h5). Dataset yang digunakan adalah dataset Iris dan model Random Forest serta neural network (TensorFlow). Dari hasil yang terlihat pada gambar, Pickle sangat cepat dalam menyimpan data dibandingkan metode serialisasi lainnya, begitu juga dengan kecepatan dalam memuat data. ONNX lebih unggul dalam ukuran penyimpanan karena lebih kecil daripada metode lainnya.

Untuk metode joblib, metode ini menghasilkan ukuran file paling besar dengan yang lainnya. Joblib memang tidak hanya meyimpan struktur model dan weight saja tetapi juga disimpan dengan metadatanya. Untuk HDF5, metode ini memerlukan waktu yang cukup lama dibandingkan metode lainnya karena struktur file yang hierarkis (kompleks, terorganisir seperti direktori). Dari hasil tersebut, dapat disimpulkan bahwa setiap metode serialisasi memiliki keunggulan masing-masing, apabila ingin ukuran file yang kecil dan mendukung bekerja lintas platform maka dapat digunakan ONNX, jika ingin menyimpan dan memuat data dengan sangat cepat dapat digunakan Pickle, lalu joblib jika ingin menyimpan data yang lengkap tidak hanya struktur model saja, dan HDF5 (.h5) apabila ingin menyimpan model kecil TensorFlow.