Modul Praktikum

Kecerdasan Buatan

9

Daftar Isi

Data Pipeline 2
TensorFlow Dataset 2
Import Datasets 2
- Local Dataset 2
  - Tabular 2
  - Text 3
  - Image 4
- Remote Dataset 5
- TFDS Dataset 6

ETL 7
- Extract 7
  - TFDS Dataset 8
  - Dataset Info 8
  - Spliting dengan tfds 9
  - Explore Split API 9
- Transform 10
- Load 10

TensorFlow – Data Pipeline

Data Pipeline

Data pipeline adalah cara memindahkan data dari suatu tempat (source) ke tempat tujuan (data warehouse). Data pipeline memungkinkan ekstraksi data dari banyak sumber yang berbeda kemudian mentransformasikan dan menggabungkannya dalam satu tempat penyimpanan data.

TensorFlow Dataset

TensorFlow Datasets adalah sekumpulan dataset yang siap pakai, seperti TensorFlow atau framework ML Python lain seperti Jax. Semua dataset dapat diakses dengan tf.data.Datasets . Memungkinkan penggunaan yang mudah dan input pipeline berperforma tinggi.

Import Datasets

Terdapat beberapa cara untuk mengambil dataset. Berikut merupakan beberapa cara yang dapat digunakan untuk mengambil dataset berdasarkan tempat dataset

- **Local Dataset**
  Mengambil dataset secara local.
  - **Tabular**
    Dataset dalam bentuk tabel. Biasanya memiliki tipe file xlsx atau csv.

    dataset

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OrdinalEncoder

dataset_raw = pd.read_csv('tested.csv')
dataset_raw.head(5)

x_train,            x_test,            y_train,            y_test            =
train_test_split(dataset_raw[['Pclass','Sex','Age','SibSp','Parch','Fare']],
dataset_raw['Survived'], test_size=.20)

encoder = OrdinalEncoder()
x_train['Sex'].shape
```

```
x_train['Sex'] = encoder.fit_transform(x_train[['Sex']])
x_test['Sex'] = encoder.fit_transform(x_test[['Sex']])

x_train = np.asarray(x_train)
x_test = np.asarray(x_test)
y_train = np.asarray(y_train)
y_test = np.asarray(y_test)

model = tf.keras.Sequential([
    tf.keras.layers.Dense(32, activation='relu', input_shape=[6]),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(
    optimizer = 'adam',
    loss = 'binary_focal_crossentropy',
    metrics = ['accuracy']
)

model.fit(
    x_train, y_train,
    epochs=10
)
```

- **Text**
  Dataset dalam bentuk teks atau kalimat. Biasanya memiliki tipe file csv atau txt.

  dataset

```
import urllib
import json

data_url                                                                    =
'https://github.com/dicodingacademy/assets/raw/main/Simulation/machine_learnin
g/sarcasm.json'
    urllib.request.urlretrieve(data_url, 'sarcasm.json')

    with open("sarcasm.json", 'r') as file:
        dataset = json.load(file)
```

- **Image**
  Dataset dalam bentuk gambar. Biasanya memiliki tipe file jpeg atau png.

```
import os
import shutil
import re
import random

def dataset_review(dataset_type):
    """Checking what directory exist in datataset directory

    Args:
        dataset_type (string): root path for choosen dataset

    Returns:
        dictionary: store dictionary name as key and sum of image in it
    """
    dataset_path = os.path.join(path_source, dataset_type)
    folders = {}

    for folder in os.listdir(dataset_path):
        folders[folder] = len(os.listdir(os.path.join(dataset_path, folder)))

    return folders

datasetDict = dataset_review(dataset_robot)

temp = 0
for key in datasetDict:
    temp += datasetDict[key]

print(f"There are {len(datasetDict)} classes with {temp} images in total",
end="\n\n")
for key, value in datasetDict.items():
    print(f"class {key} have {value} images")

def create_dir(rooth_path, folders):
    """Create the directory for later be the place to store a suitable dataset

    Args:
        rooth_path (string): the root path where directory will be created
          folders (dict): dictionary that store label name and the summary of
images inside it
    """
```

```
    if os.path.isdir(os.path.join(rooth_path, "training")):
            print(f"this  path  {os.path.join(rooth_path,  'training')}  already
made")
    else:
        os.makedirs(os.path.join(rooth_path, "training"))
        for folder in folders.keys():
            os.makedirs(os.path.join(rooth_path, f"training/{folder}"))

    if os.path.isdir(os.path.join(rooth_path, "validation")):
           print(f"this  path  {os.path.join(rooth_path,  'validation')}  already
made")
    else:
        os.makedirs(os.path.join(rooth_path, "validation"))
        for folder in folders.keys():
            os.makedirs(os.path.join(rooth_path, f"validation/{folder}"))
```

- **Remote Dataset**

  Mengambil dataset secara remote atau dari internet.

```
import urllib.request

data_url                                                                    =
'https://github.com/dicodingacademy/assets/releases/download/release-rps/rps.z
ip'
    urllib.request.urlretrieve(data_url, 'rps.zip')
    local_file = 'rps.zip'
    zip_ref = zipfile.ZipFile(local_file, 'r')
    zip_ref.extractall('data_b3/')
    zip_ref.close()
```

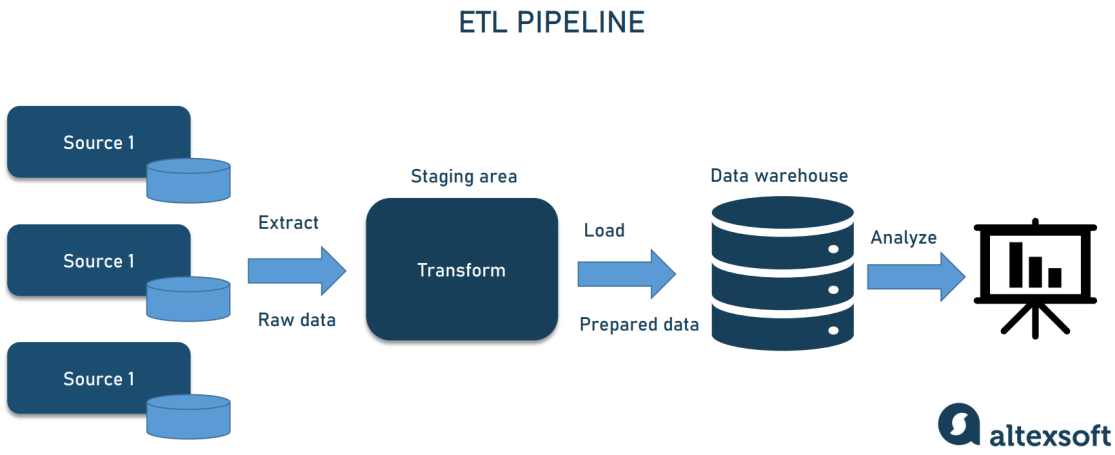
atau

```
import pandas as pd

bbc = pd.read_csv(

'https://github.com/dicodingacademy/assets/raw/main/Simulation/machine_learnin
g/bbc-text.csv')
```

- **TFDS Dataset**

  tensorflow_datasets (tfds) merupakan API yang sering digunakan untuk mengambil dataset dalam TensorFlow. Dengan tfds kita bisa menggunakan dataset tensorflow secara instan. Terutama mempermudah kita untuk memasukan dataset kedalam data pipeline kita melalui proses ETL.

ETL



ETL adalah proses ektraksi, tranformasi, dan memuat data dari beberapa sumber ke data warehouse atau penyimpanan data terpadu. Proses ini menyediakan pondasi untuk alur kerja data analisis dan machine learning. ETL membersihkan dan mengatur data dengan memetakan kebutuhan bisnis tertentu seperti laporan bulanan, tetapi dapat digunakan juga untuk menyelesaikan analisis yang kompleks. Hal ini dapat meningkatkan proses di bagian back-end.

- **Extract**
  Ekstraksi adalah proses kita mengekstrak atau memuat suatu dataset kedalam projek. kita bisa mengekstrak data menggunakan **tfds.load**

  adapun parameter yang bisa kita gunakan seperti :

  - name : menentukan nama dataset yang akan di load
  - split : untuk menentukan tipe splitting
  - as_supervised : memberikan feature dengan label
  - with_info : memuat informasi metadata
  - data_dir : keterangan direktori untuk read/write
  - batch_size : menambah batch dimension sebagai contoh
  - shuffle_files : untuk mengacak input file
  - download : untuk mendownload kedalam direktori

Contoh :

```
import tensorflow_dataset as tfds

datasets = tfds.load(name='mnist', split='train')
```

TFDS Dataset

Ada banyak dataset tensorflow yang dapat digunakan. Kita bisa melihat daftar dataset dengan fungsi **tfds.list_builders()**

```
import tensorflow_datasets as tfds

dataset = list(tfds.list_builders)

for data in dataset:
    print(data)
```

Dataset Info

Untuk melihat detail dari dataset tfds yang akan digunakan, kita bisa menggunakan paramater with_info.

```
import tensorflow_dataset as tfds

datasets, info = tfds.load(name='mnist', with_info)

print(info)
```

Kita juga dapat memilih info detail dari contoh diatas seperti berikut.

```
print("URL Datasets : ", info.urls)
print("Images Features : ", info.features['image'])
print("Label Features : ", info.features['label'])
print("Jumlah data training : ", info.split['train'].num_examples)
print("Jumlah data testing : ", info.split['test'].num_examples)
```

Spliting dengan tfds

Saat mengekstraksi dataset, kita dapat mengatur pembagian dataset pada parameter split. Ada beragam cara untuk melakukan spliting saat load dataset.

tfds.Split :

- TRAIN

```
tfds.load(name_dataset, split=tf.Split.TRAIN)
```

- VALIDATION

```
tfds.load(name_dataset, split=tf.Split.VALIDATION)
```

- TEST

```
tfds.load(name_dataset, split=tf.Split.TEST)
```

- ALL

```
tfds.load(name_dataset, split=tf.Split.ALL)
```

Explore Split API

Selain memilih jenis data split, kita juga bisa meng-custom spliting data yang kita inginkan. Berikut implementasinya.

- **Split biasa**

```
train, test = tfds.load('mnist', split=['train','test']

print(len(list(train)))
print(len(list(test)))
```

- **Kombinasi**

```
kombinasi = tfds.load('mnist', split='train+test')

print(len(list(kombinasi)))
```

- **Mengambil jumlah spesifik**

```
10k_pertama = tfds.load('mnist', split='train[:10000]')

print(len(list(10k_pertama)))
```

- **Mengambil jumlah persentase**

```
20persen_pertama = tfds.load('mnist', split='train[:20%]')

print(len(list(20persen_pertama)))
```

- **Susunan**

```
susunan = tfds.load('mnist', split='test[:10%]+train[-80%]')

print(len(list(susunan)))
```

- **Transform**
  Transformasi berguna untuk mengubah, memanipuliasi, atau mentransformasikan dataset sebelum digunakan kedalam model. Ada banyak cara untuk mentransformasikan dataset. Berikut beberapa contoh untuk mentransform dataset.

```
dataset = dataset.shuffle(NUM_SAMPLES)
dataset = dataset.repeat(NUM_EPOCHS)
dataset = dataset.map(lambda x:..)
dataset = dataset.batch(BATCH_SIZE)
```

- **Load**
  Sedangkan Load sendiri ialah proses mengambil dataset yang telah diekstrak agar siap diinputkan kedalam model.

```
for data in dataset.take(1):
    image = data["image"].numpy().squeeze()
    label = data["label"].numpy()

    print("Label: {}".format(label))
    plt.imshow(image, cmap=plt.cm.binary)
    plt.show()
```