Catatan dari penulis:
Hasil pemikiran tidak dipublikasikan. Terdokumentasi di Perpustakaan Departemen Ilmu Komputer IPB, untuk pembimbingan tugas akhir mahasiswa.
Koresponding: endang_pg@apps.ipb.ac.id

PENGANTAR PEMROGRAMAN DENGAN PYTHON UNTUK PENELITIAN MENGGUNAKAN ANACONDA DAN JUPYTER NOTEBOOK

ENDANG PURNAMA GIRI, SONY HARTONO WIJAYA & TOTO HARYANTO
Staf Pengajar: Departemen Ilmu Komputer FMIPA IPB

DEPARTEMEN ILMU KOMPUTER
INSTITUT PERTANIAN BOGOR
BOGOR
2023

RINGKASAN

ENDANG PURNAMA GIRI, SONY HARTONO WIJAYA, dan TOTO HARYANTO. Staf Pengajar Departemen Ilmu Komputer IPB, Fakultas Matematika dan Ilmu Pengetahuan Alam, IPB.

Kemampuan pemrograman merupakan kompetensi utama mahasiswa ilmu Komputer dan teknik infomatika. Python merupakan salah satu bahasa pemrograman yang banyak digunakan pada beragam penelitian. Hal tersebut karena banyaknya library pendukung pada lingkungan pemrograman python. Buku ini merupakan buku yang ditujukan sebagai pendamping bagi mahasiswa yang ingin mempelajari pemrograman dengan python. Lebih lanjut melalui buku ini mahasiswa yang akan melakukan penelitian dan menggunakan python sebagai Bahasa pemrograman bisa memperoleh referensi yang mudah dipahami untuk menuliskan kode programnya.

Kata kunci: pemrograman, python, jupyter

Catatan dari penulis:
Hasil pemikiran tidak dipublikasikan. Terdokumentasi di Perpustakaan Departemen Ilmu Komputer IPB, untuk pembimbingan tugas akhir mahasiswa.
Koresponding: endang_pg@apps.ipb.ac.id

DAFTAR ISI

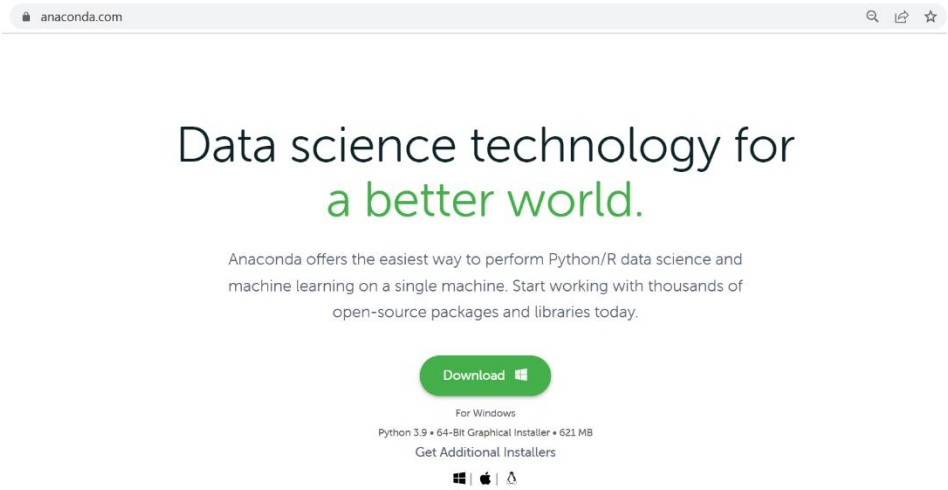
DAFTAR GAMBAR

Pernyataan Penulis: Hasil pemikiran, tidak dipublikasikan dan disimpan di Perpustakaan Departemen Ilmu Komputer FMIPA IPB, untuk bahan

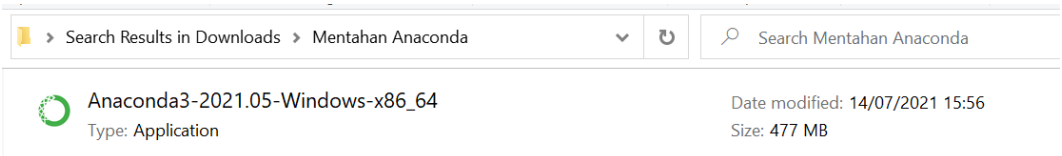
1 INSTALASI LINGKUNGAN PEMROGRAMAN PYTHON MENGGUNAKAN ANACONDA

Agar komputer kita dapat menginterpretasikan kode program python yang kita tulis maka tentunya komputer kita harus sudah terinstallkan interpreter python. Pada bagian ini akan dipaparkan cara menginstalasi lingkungan python pada framework Anaconda. Pemilihan Anaconda sendiri untuk menjadikan lingkungan python yang kita gunakan menjadi lebih mudah dalam pengelolaan librarynya. Untuk mendukung kebutuhan ini akan diperkenalkan konsep environment. Sederhananya menggunakan environment ini kita dapat membuat beberapa lingkungan terpisah dan berbeda pada suatu paket versi python ataupun versi library yang memang kita gunakan. Misalkan pada environment pertama kita butuh dan menggunakan python versi 2.7 sedangkan pada environment kedua kita gunakan python versi 3.8. Hal tersebut mungkin dilakukan dan tidak akan saling mengganggu ketika terjadi update instalasi library dilakukan pada masing-masing environment. Hal ini karena masing-masing environment memiliki lingkungan tersendiri dan tidak saling terhubung.

Pengunduhan Material Anaconda



Gambar 1 Lokasi material instalasi Anaconda.

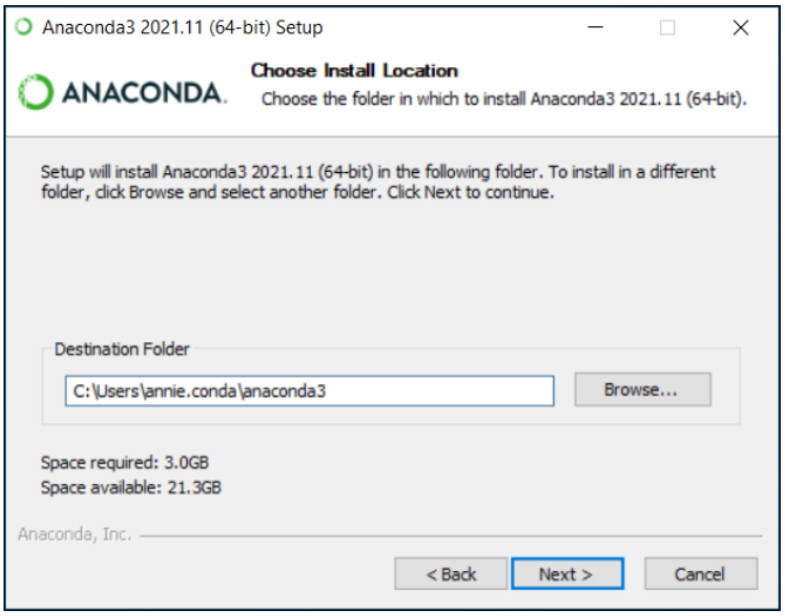


Gambar 2 Material instalasi Anaconda3 yang digunakan.

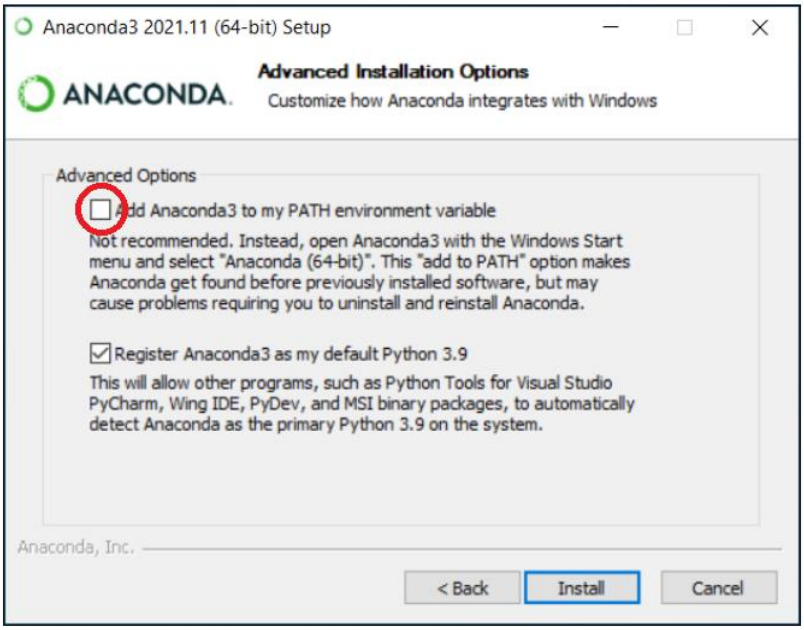
Langkah pertama yang harus kita lakukan adalah menyiapkan materi instalasi Anaconda yang dapat di download pada https://www.anaconda.com/ (Gambar 1).

Pada dokumen ini sistem operasi yang digunakan adalah Windows (tepatnya Windows 10) 64 Bit. Pastikan material instalasi Anaconda yang kita download sesuai dengan system operasi yang kita gunakan.

Instalasi Anaconda



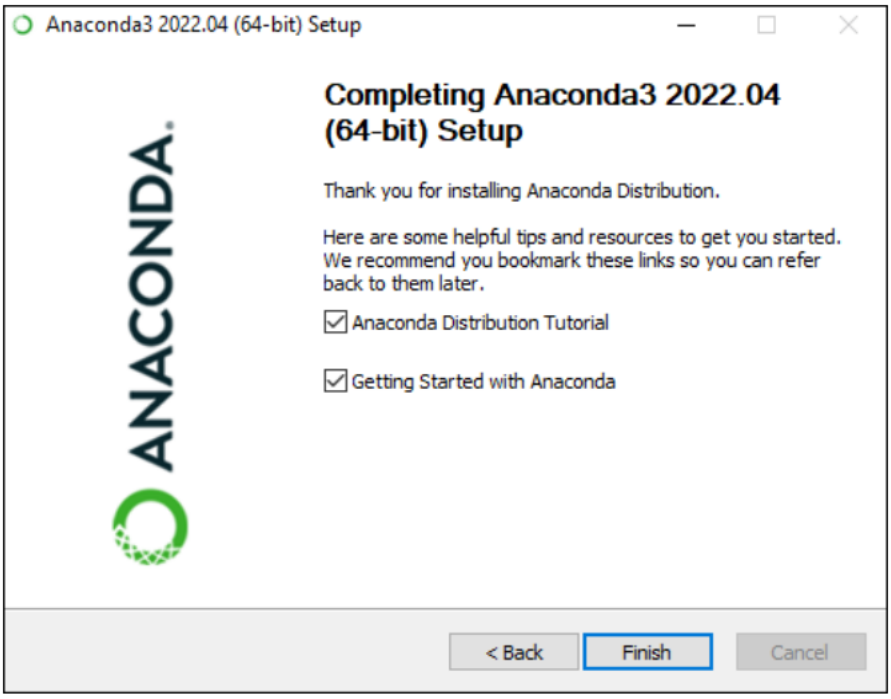
Gambar 3 Layar konfirmasi instalasi Anaconda.



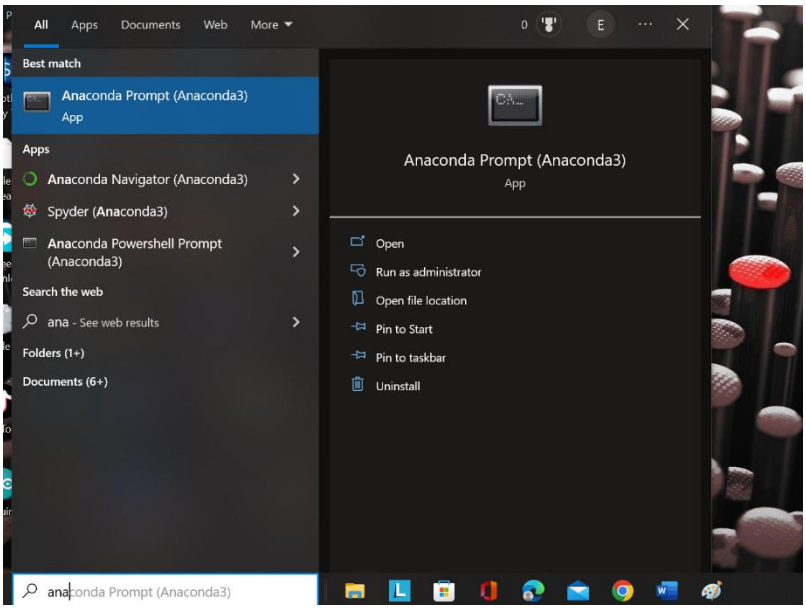
Gambar 4 konfirmasi instalasi menambahkan pada PATH environment variable.

Anaconda yang digunakan pada dokumen ini adalah Anaconda Versi3 (Gambar 2). Klik dua kali pada material instalasi Anaconda yang sudah kita unduh. Silahkan tentukan folder tujuan tempat Anaconda akan diinstall dan pastikan ruang hard disk komputer kita cukup besar sesuai dengan yang dibutuhkan pada saat

konfirmasi instalasi akan di dilakukan (Gambar 3). Salah satu yang jangan terlupakan pastikan pilihan menambahkan Anaconda kepada PATH environment variable system operasi windows kita harus terceklis (terpilih) (Gambar 4). Jika proses instalasi berjalan baik maka akan kita peroleh layer konfirmasi instalasi seperti pada Gambar 5.



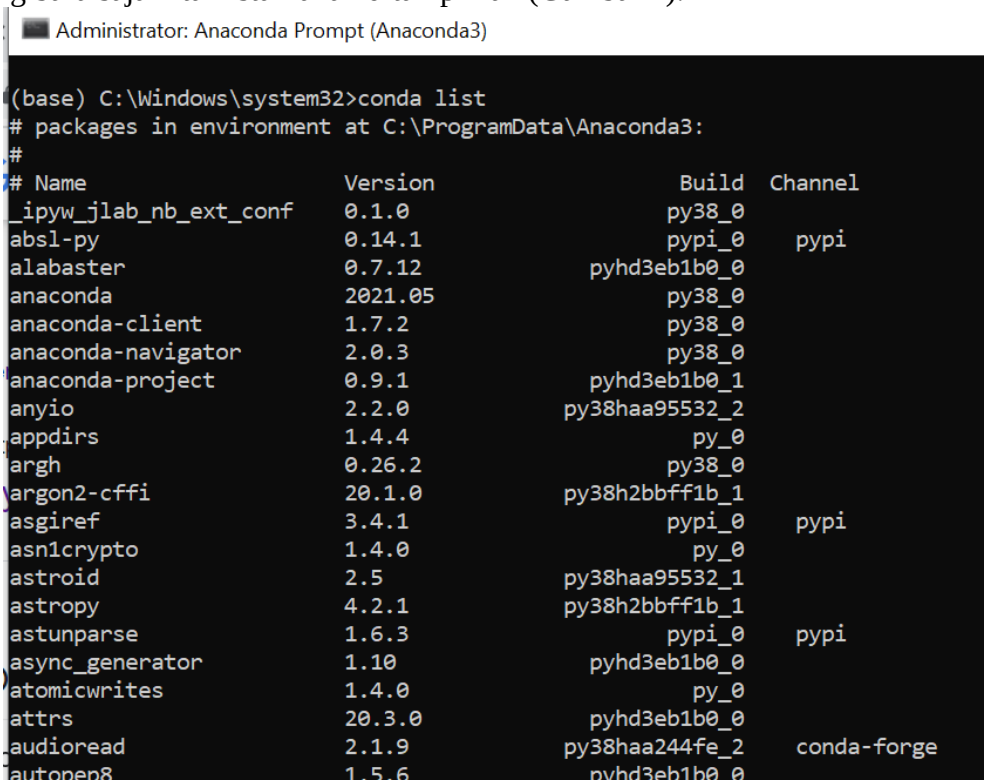
Gambar 5 Konfirmasi selesainya proses instalasi.



Gambar 6 Beragam aplikasi hasil install Anaconda pada komputer kita.

Sebagai hasilnya saat ini Anaconda sudah terinstall pada komputer kita. Ketika kita melakukan penelusuran pada kolom search maka akan kita temukan

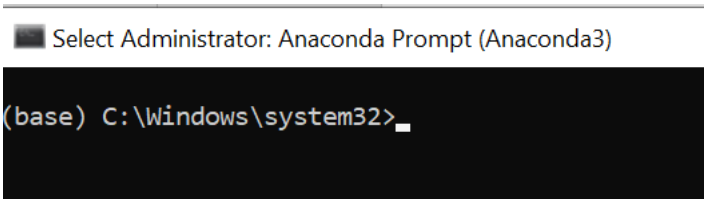
seperangkat aplikasi seperti pada Gambar 6 (Anaconda Prompt, Anaconda Navigator, Spyder, dan Anaconda Powershell prompt). Untuk mengecek package Anaconda yang baru saja kita install, bisa kita jalankan anaconda prompt lalu run perintah “conda list”. Menggunakan perintah ini maka beragam package Anaconda yang baru saja kita install akan ditampilkan (Gambar 7).



Gambar 7 Daftar package yang telah kita install.

Konsep Environment pada Anaconda

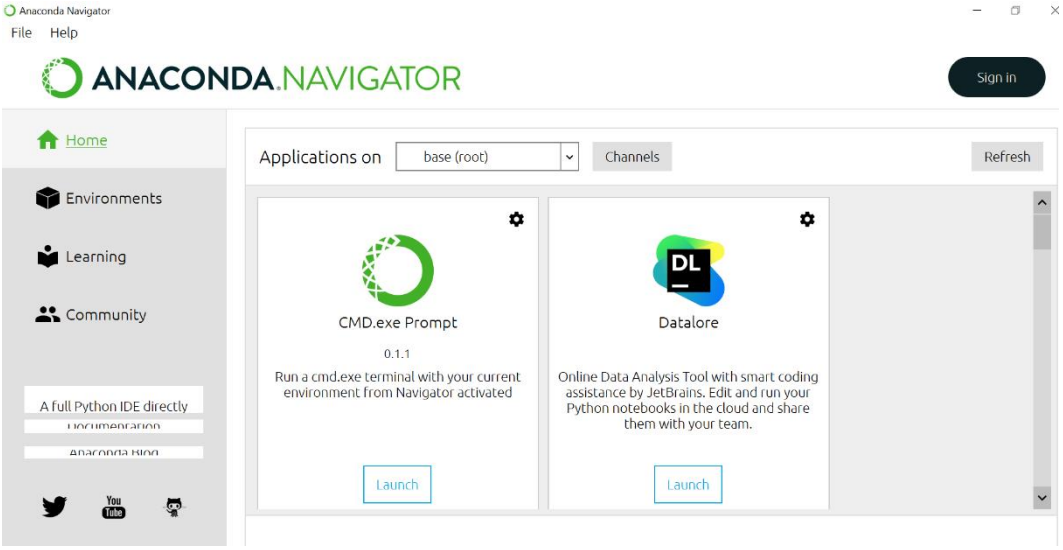
Sebelum kita lanjutkan untuk instalasi Jupyter Notebook sebagai IDE yang akan kita gunakan maka akan dipaparkan konsep environment pada lingkungan Anaconda. Cara ini bisa kita lakukan dengan terlebih dahulu masuk ke Anaconda Prompt. Jalankan atau Run as Administrator supaya kita memiliki kewenangan penuh terhadap lingkungan system computer kita (utamanya ketika akan menginstall library tambahan pada lingkungan python kita).



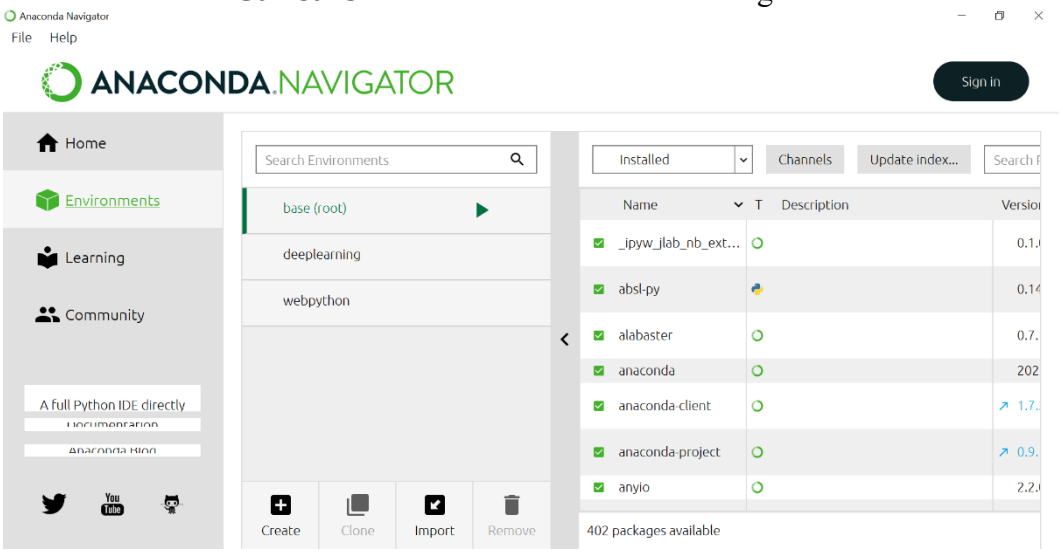
Gambar 8 Jendela Anaconda Prompt.

Tampilan awal ketika kita menjalankan Anaconda Prompt akan kita peroleh jendela aplikasi seperti pada Gambar 8. Pada kondisi ini kita akan secara default pada environment base. Untuk membuat sebuah environment baru maka cara yang

paling mudah bisa kita jalankan Anaconda Navigator (Gambar 9). Selanjutnya arahkan pada menu Environments (Gambar 10). Pada menu ini akan dapat kita lihat beragam environment python yang telah ada di computer kita. Pada Gambar 10 terlihat ada tiga buah environment yang telah ada di computer yaitu base, deeplearning, dan webpython. Environment base merupakan environment default yang otomatis sudah ada ketika pertama kali kita install Anaconda. Adapun dua environment lainnya adalah environment yang telah dibuat sebelumnya secara mandiri.



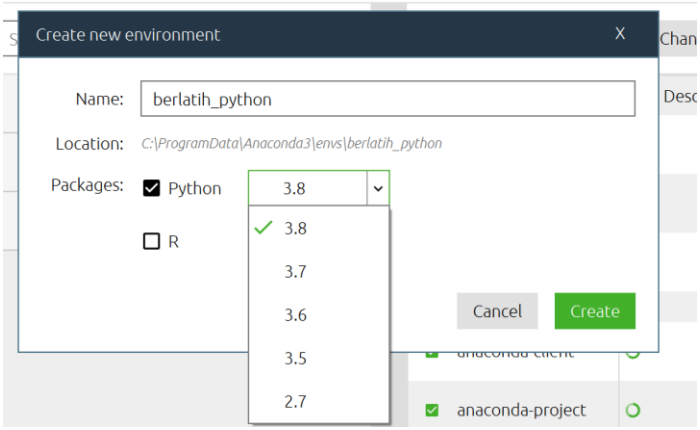
Gambar 9 Jendela Home Anaconda Navigator.



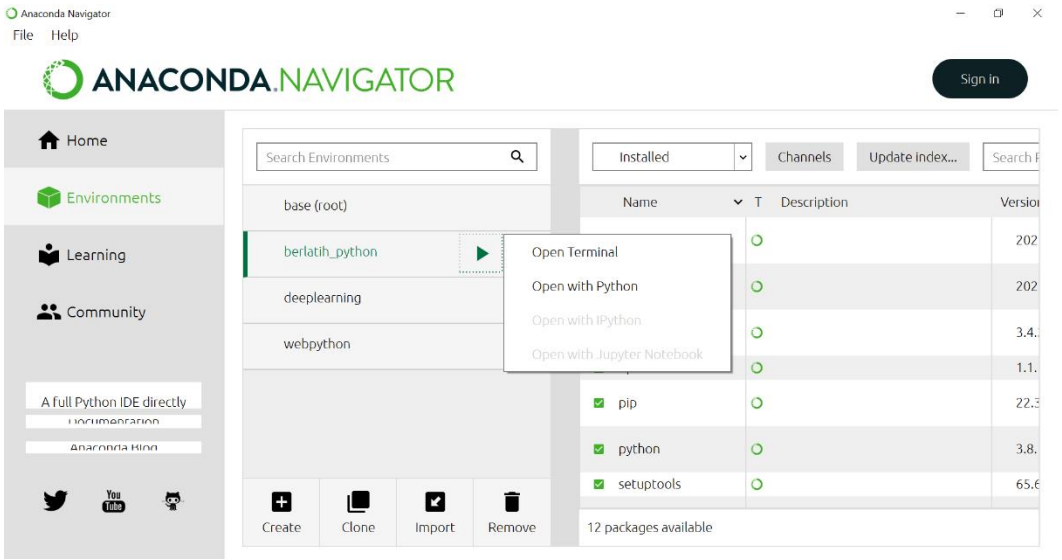
Gambar 10 Jendela Environments Anaconda Navigator.

Untuk membuat environment baru maka kita bisa klik tombol create pada bagian bawah dari tab Environments ini (Gambar 11). Saat membuat environment baru harus kita tentukan nama environment dan lingkungan Python versi berapa yang akan digunakan untuk environment baru yang kita buat ini. Pada ilustrasi ini kita beri nama environmentnya adalah berlatih_python sedangkan lingkungan Python yang kita gunakan adalah Python versi 3.8. Setelah kita membuat sebuah environment maka environment yang kita buat tersebut akan dicantumkan pada

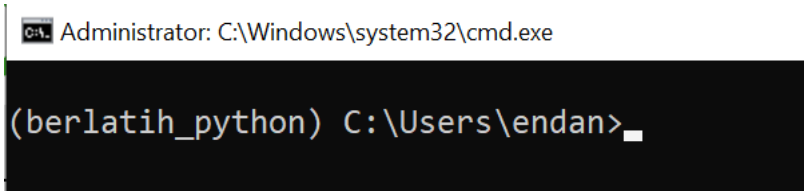
daftar environment yang ada pada Anaconda System dari computer kita. Untuk menjalankan suatu environment yang kita miliki maka tinggal klik pilih open terminal (Gambar 12). Sebagai hasilnya maka kita akan membuka jendela terminal dari environment yang kita jalankan (Gambar 13).



Gambar 11 Aksi membuat environment baru.



Gambar 12 Menjalankan sebuah environment dari Anaconda Navigator.



Gambar 13 Terminal environment berlatih_python

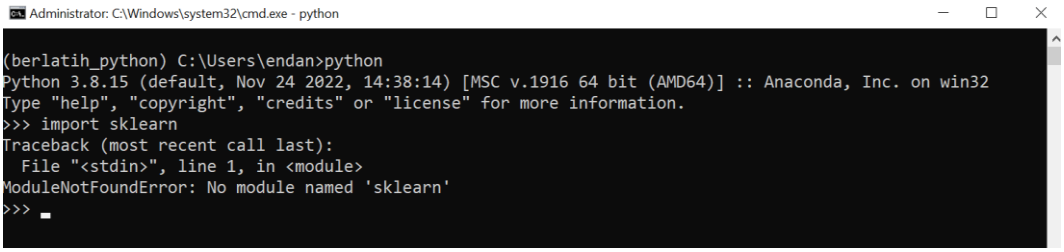
Menginstall Library Baru pada Sebuah Environment

Seperti telah disebutkan sebelumnya sebuah environment pada Anaconda adalah suatu lingkungan pemrograman python yang independent terhadap environment lainnya. Oleh karena itu kita dapat secara bebas mengkonfigurasi dan

menginstallkan library yang kita butuhkan pada sebuah environment. Langkah yang harus dipastikan adalah kita telah berada pada environment yang memang ingin kita tuju. Pada Gambar 13 menyatakan bahwa saat ini kita berada pada environment dengan nama “berlatih_python”. Pada saat sebuah environment dibuat bisa jadi lingkungan pemrograman kita pada environment tersebut tidak memiliki banyak library. Pada Gambar 14 diilustrasikan bagaimana kita memasuki mode python (mengetikan python) untuk kemudian kita mencoba mengimport library sklearn. Sebagai hasilnya maka kita mengalami kegagalan melakukan aksi ini. Hal tersebut dikarenakan library sklearn belum terinstall pada environment berlatih_python.

Kode instruksi:

- Memasuki mode python: python
- Mengimport sklearn: import sklearn

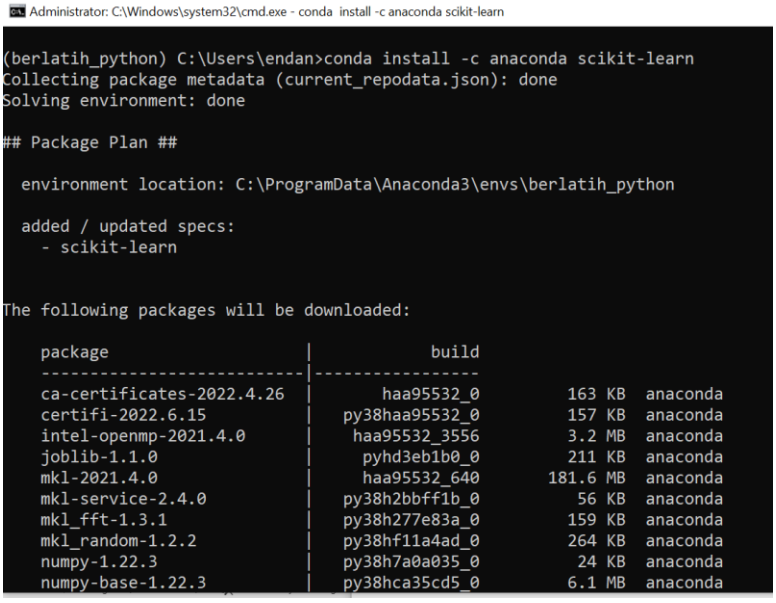


Gambar 14 Kegagalan mengimport sebuah library

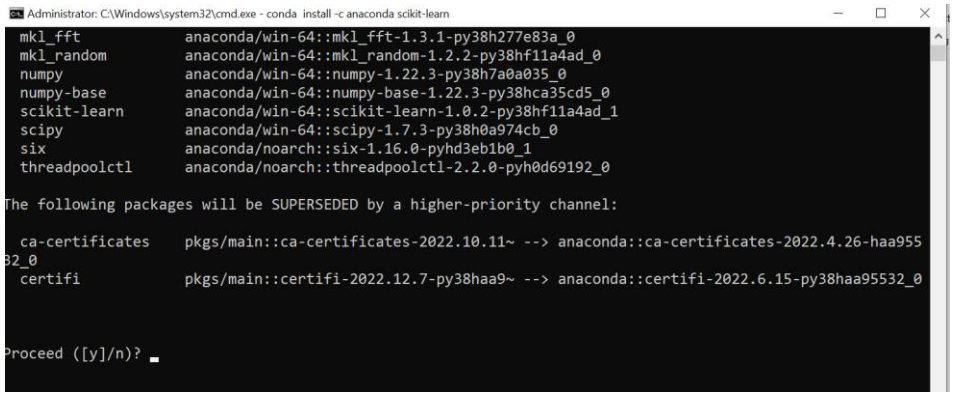
Untuk mengatasi ini maka kita akan kembali menuju mode utama dari environment (keluar dari mode python) untuk kemudian menginstallkan library yang sebelumnya gagal kita import (sklearn).

Kode instruksi:

- Dari mode python ketikan exit() untuk keluar dari mode python
- Menginstall libraru sklearn: conda install -c anaconda scikit-learn

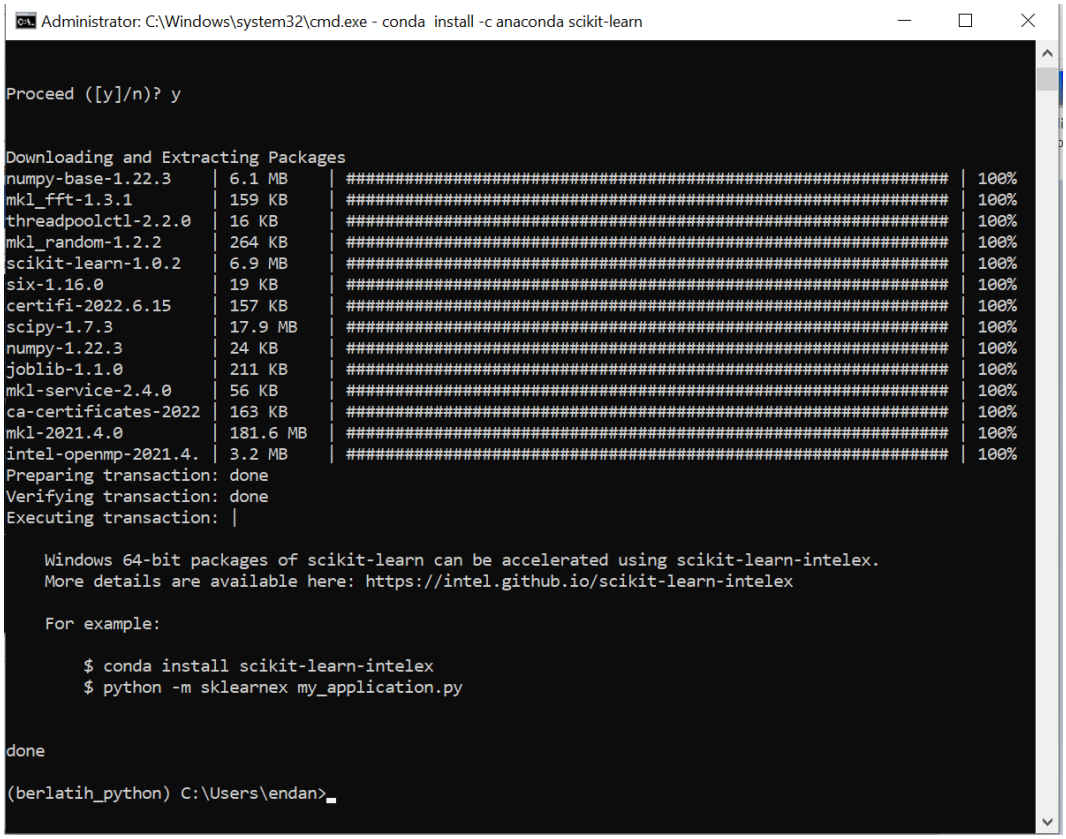


Gambar 15 Terminal environment berlatih_python.



Gambar 16 Terminal environment berlatih_python.

Setelah kita ketikkan instruksi untuk menginstall library sklearn maka akan kita peroleh tampilan seperti pada Gambar 15. Untuk dapat melakukan ini pastikan mode pembukaan terminal kita sebagai administrator (Run as Administrator) dan pastikan tersedianya koneksi internet. Koneksi internet akan digunakan untuk mendownload library yang sesuai dengan system lingkungan kita dari repository. Pada bagian akhir dari tampilan ini akan diminta konfirmasi apakah proses instalasi package library akan dilakukan? Ketikkan “y” untuk melanjutkan proses instalasi package. Hasil akhir pengunduhan library dan installasi diberikan seperti pada Gambar 17.



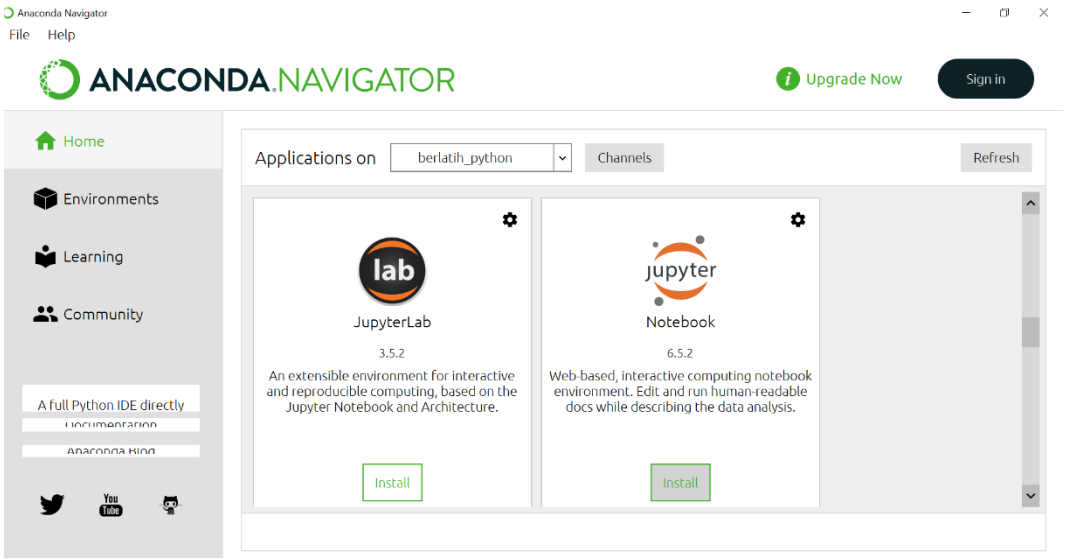
Gambar 17 Tampilan terminal dari prosedur installasi package library.

Untuk mengujicoba apakah library sklearn saat ini telah tersedia pada environment kita maka kita lakukan prosedur import sklearn seperti sebelumnya. Seperti terlihat pada Gambar 18, sekarang kita tidak mengalami kegagalan dalam proses import sklearn. Hal ini mengindikasikan bahwa saat ini environment kita telah memiliki library sklearn. Dengan metode yang sama dengan yang dijelaskan barusan maka secara mudah kita bisa menginstall beragam library lainnya pada environment yang kita miliki.

```
(berlatih_python) C:\Users\endan>python
Python 3.8.15 (default, Nov 24 2022, 14:38:14) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sklearn
>>>
```

Gambar 18 Tampilan terminal dari prosedur installasi package library.

2 INSTALASI JUPYTER NOTEBOOK SEBAGAI IDE



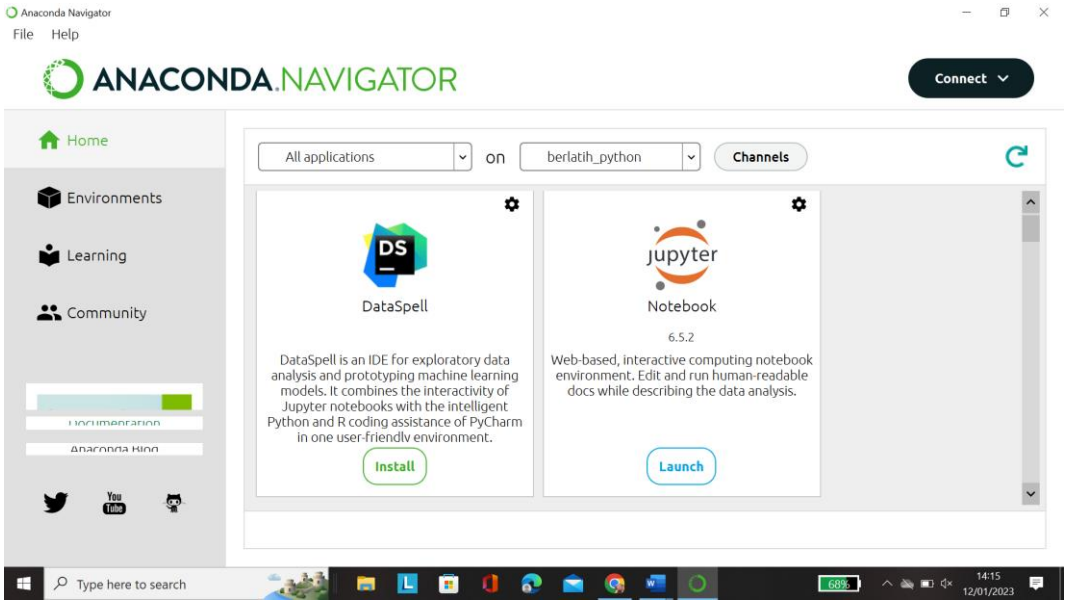
Gambar 19 Beranda home untuk library environment pada Anaconda Navigator.

Untuk Integrated Development Environment (IDE) tempat menuliskan kode program python kita pada lingkungan Anaconda sebenarnya dapat kita gunakan spyder, namun pada dokumen ini kita akan menggunakan Jupyter Notebook. Oleh karena itu maka pada bagian ini disampaikan prosedur menginstall Jupyter Notebook pada Anaconda yang telah kita install sebelumnya.

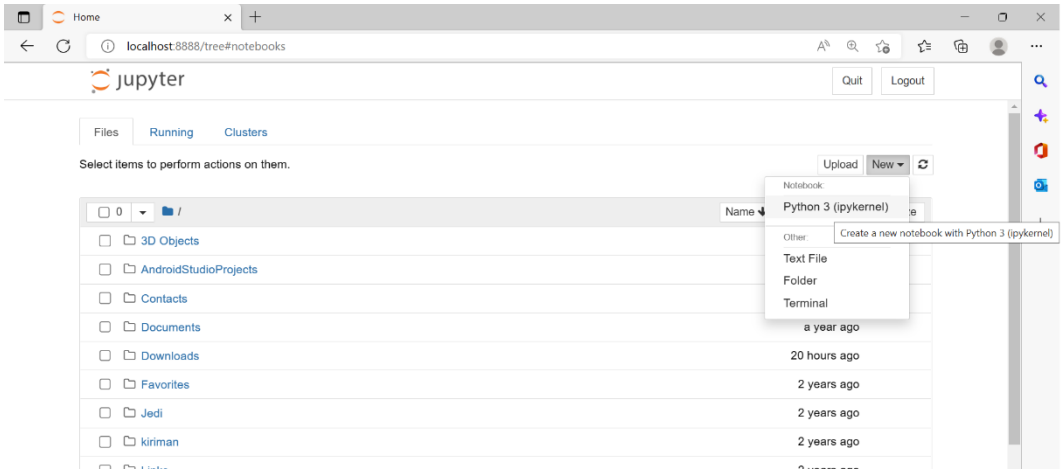
Instalasi Jupyter Notebook pada *Environment* Anaconda

Menggunakan Anaconda maka prosedur instalasi Jupyter Notebook sangat mudah dilakukan. Langkah yang kita lakukan adalah dengan menjalankan Anaconda Navigator. Pastikan aplikasi Anaconda Navigator merupakan versi terupdate agar proses instalasi dapat terdukung dan berjalan dengan lancar. Pada bagian Home pilih bagian Application on kepada environment yang ingin dituju.

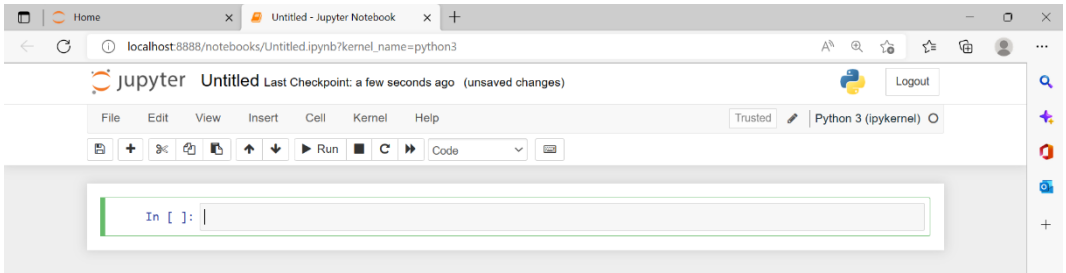
Pada contoh ini environment yang dituju bernama berlatih_python. Lalu telusuri bagian Notebook dan klik install (Gambar 19). Pastikan koneksi internet tersedia dengan baik. Ketika proses instalasi telah selesai dilakukan maka tombol install akan menjadi launch (Gambar 20). Setelah ini maka Jupyter Notebook dapat di jalankan secara langsung dengan mengklik tombol Launch.



Gambar 20 Status library Jupyter Notebook setelah terinstall.



Gambar 21 Tampilan Jupyter Notebook saat dijalankan.

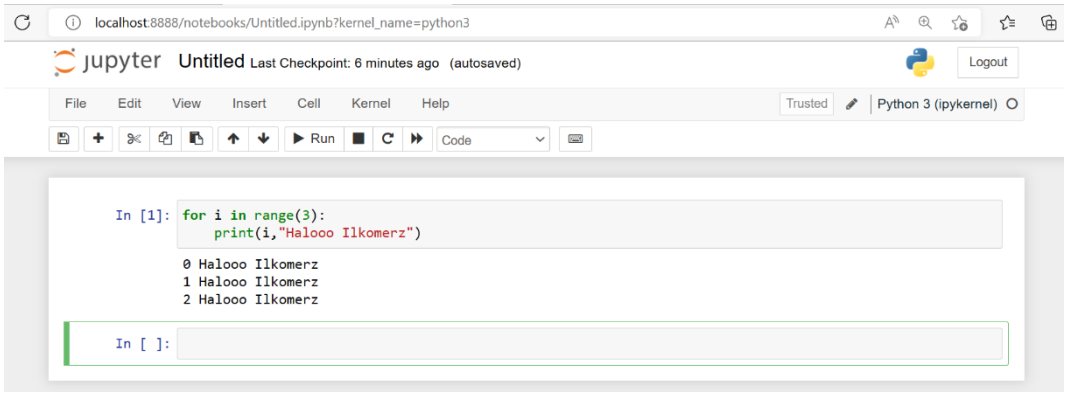


Gambar 22 Tampilan Jupyter Notebook saat dijalankan.

Untuk menjalankan Jupyter Notebook maka bisa langsung meng-klik tombol Launch pada Anaconda Navigator. Pastikan environment yang terpilih sesuai dengan yang ingin kita gunakan (berlatih_python). Jupyter Notebook akan dijalankan melalui browser (Gambar 21).

Menulis Kode Program pada Jupyter Notebook

Untuk membuka kode baru maka dapat diklik menu New dan kemudian pilihlah Python 3 (ipykernel). Sebagai hasilnya kita akan dibukakan suatu area penulisan kode program yang masih kosong (Gambar 22). Pada setiap baris input di Jupyter Notebook selanjutnya dapat kita ketikkan kode program python kita, lalu klik icon Run untuk menjalankannya. Pada Gambar 23 diilustrasikan bagaimana sebuah kode program untuk mencetak sebuah teks secara berulang. Sampai dengan titik ini maka proses instalasi Jupyter Notebook telah selesai dan siap digunakan. Alasan utama pemilihan Jupyter Notebook sebagai IDE bagi penelitian yang dituliskan menggunakan program python adalah tampilan Jupyter Notebook yang secara terintegrasi menampilkan kode program dan output dari hasil run kode. Menggunakan pendekatan ini maka kita sebagai peneliti akan lebih mudah menelusuri kode program dari penelitian yang sedang kita lakukan.



Gambar 23 Tampilan Jupyter Notebook saat dijalankan.

3 STUDI KASUS KLASIFIKASI

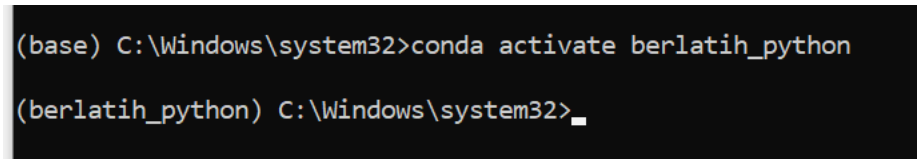
Pada Bab ketiga ini akan dipaparkan suatu contoh studi kasus penelitian sederhana di bidang klasifikasi yang dituliskan kode programnya pada Jupyter Notebook. Sebelum lebih jauh berikut ini adalah konfigurasi system yang digunakan pada kesempatan kali ini.

- **Perangkat Keras:** CPU I5 Gen 11th (8 CPU, 2,4 GHz), Memory utama 16 GB RAM, 512 Solid State Drive (SSD).
- **Perangkat Lunak:** Windows 10 Pro 64 bit, Anaconda 3, Jupyter Notebook.

Environment yang akan digunakan adalah berlatih_python. Pastikan baik menjalankan dari Anaconda Navigator maupun melalui Anaconda Prompt bahwa

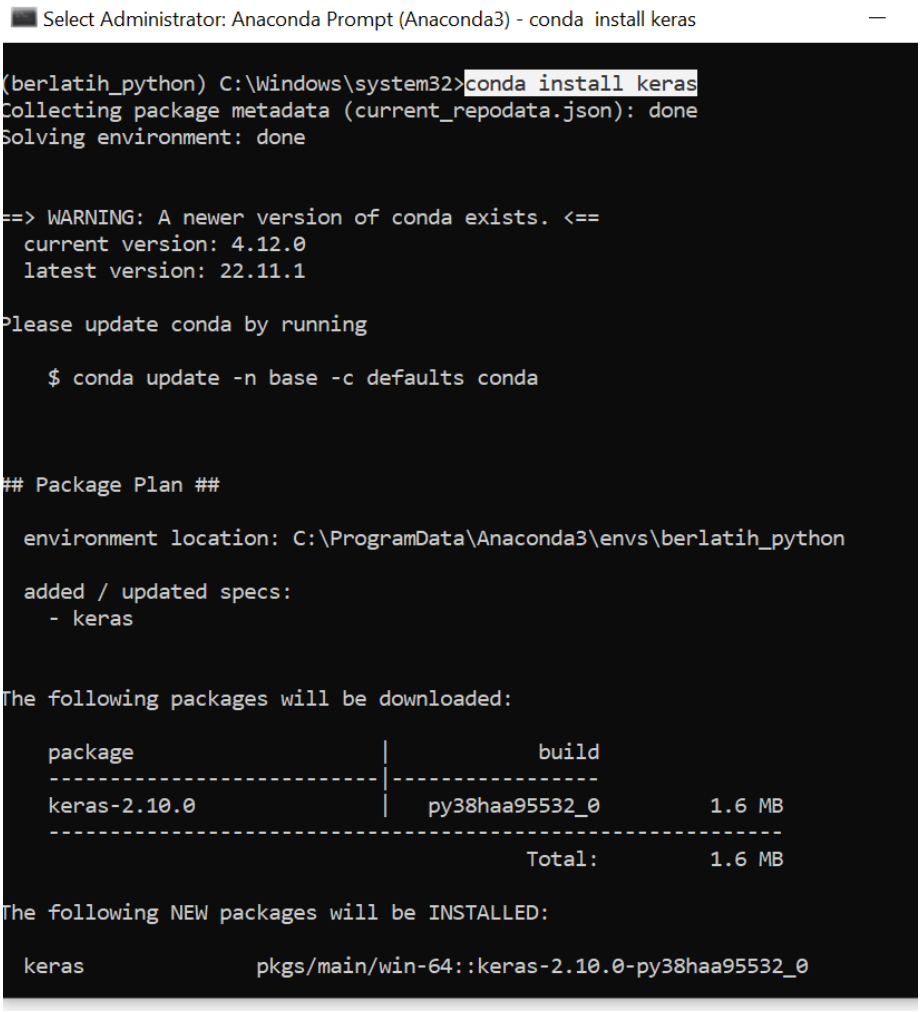
environment telah sesuai. Gambar 24 menunjukan ketika kita berpindah dari environment base sebagai default menuju environment berlatih_python.

Perintah: **conda activate** nama_env_tujuan



Gambar 24 Tampilan Jupyter Notebook saat dijalankan

Menyiapkan Beragam Library untuk Kasus Klasifikasi

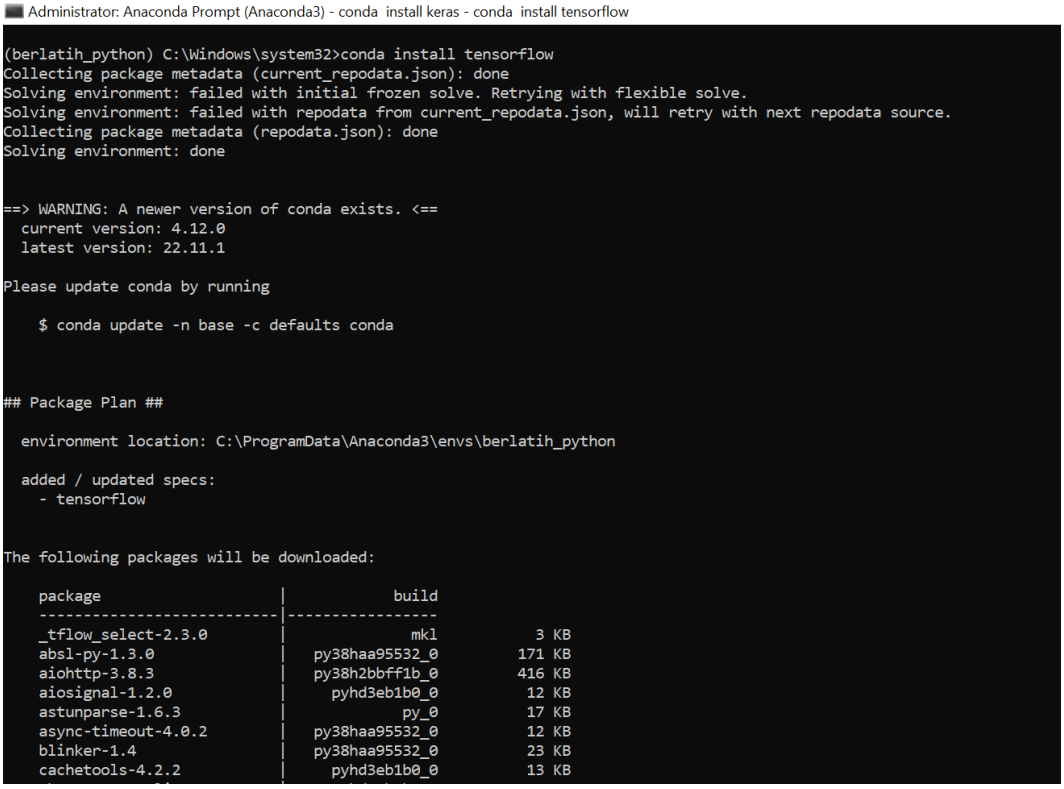


Gambar 25 Proses instalasi package Keras.

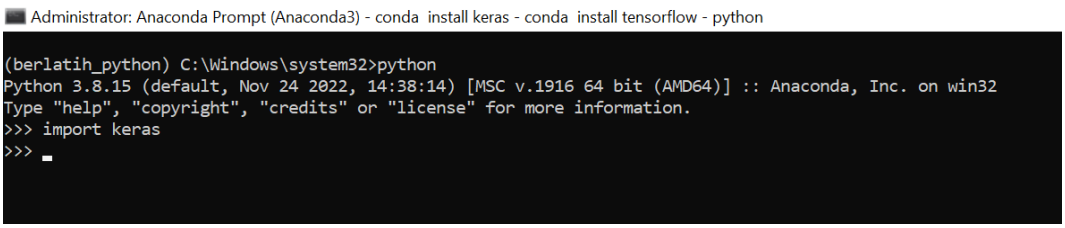
Sebagai klasifier utama pada kasus ini adalah Convolutional Neural Network dari library Keras. Pemilihan CNN sebagai klasifier didasarkan kepada beragam catatan keberhasilan kasus klasifikasi image seperti pada Ciresan *et al.* (2011), Hinton & Salakhutdinov (2006), Lecun *et al* (1998), Li (2014), Sultana *et al*. (2018). Keberhasilan CNN pada data suara dan data satu dimensi seperti pada Giri (2016) dan Hamid *et al*. Lebih lanjut klasifikasi multi channel pada CNN juga telah

memperoleh keberhasilan seperti pada penelitian Zheng *et al* (2014). Oleh karena itu kita harus menginstall *package* library Keras pada environment kita. Untuk instalasi Keras bisa kita gunakan perintah: conda install keras.

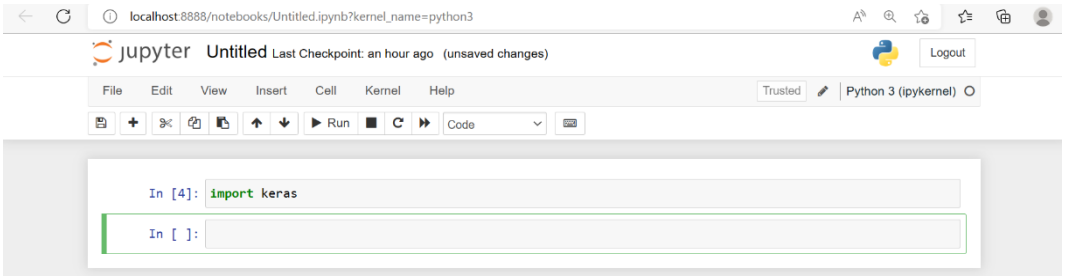
Perlu kita ketahui bersama bahwa untuk dapat menjalankan Keras maka kita juga perlu memiliki Tensor flow sebagai backend dari berjalannya Keras. Oleh karena itu maka kita juga wajib menginstallkan package Tensorflow pada environment kita (Gambar 26). Untuk melakukan instalasi tensorflow maka dapat digunakan perintah: conda install tensorflow. Perlu diingat untuk setiap proses instalasi maka computer kita harus memiliki akse internet. Hal tersebut karena proses instalasi membutuhkan prosedur download dari sumber repositori.



Gambar 26 Proses instalasi package Tensorflow.



Gambar 27 Uji mengimport package Keras.



Gambar 28 Uji mengimport package Keras melalui Jupyter Notebook.

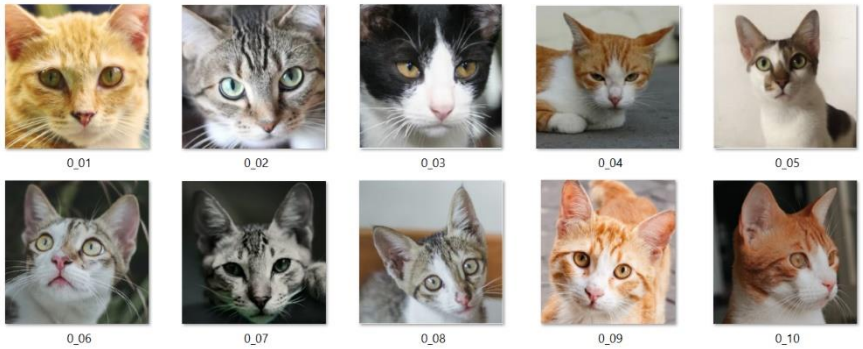
Setelah selesai melakukan prosedur instalasi bagi Keras dan Tensorflow maka untuk mengujinya kita dapat masuk ke mode python, lalu mencobakan mengimport keras. Jika tidak terjadi pesan error akibat kegagalan mengimport maka artinya environment kita telah sukses diinstallkan package Keras (Gambar 27). Uji melakukan aksi import keras juga dapat secara langsung pada Jupyter Notebook (Gambar 28).

Menyiapkan Dataset

Dataset yang digunakan pada uji kasus klasifikasi ini terinspirasi dari tulisan dengan judul 7 Jenis Kucing yang popular dipelihara di Indonesia (idntimes.com, https://www.idntimes.com/science/discovery/0044-asyiatul-husnia/jenis-kucing-yang-populer-dipelihara-di-indonesia-c1c2-1?page=all) Gambar 29. Mengacu kepada artikel ketujuh kelas kucing tersebut adalah: 1. kucing kampung (*domestic cat*); 2. kucing Persia; 3. kucing Birman; 4.kucing Siam; 5. Kucing Anggora; 6. Kucing Sphinx; dan 7. Kucing British Shorthair.

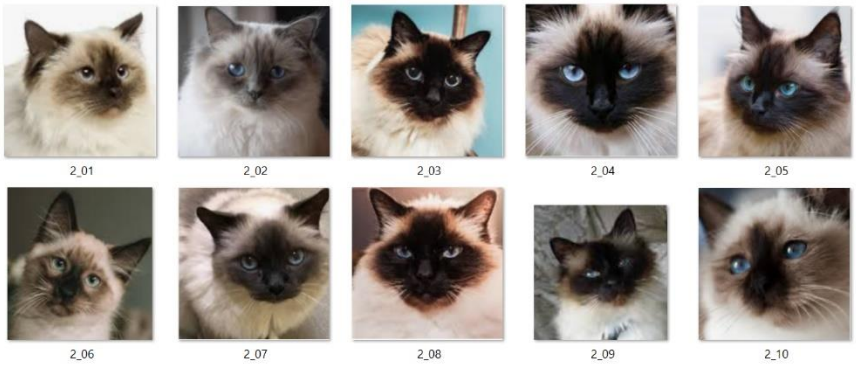
Gambar 29 Artikel klasifikasi kucing pada IDNTimes.com.

Studi kasus kali ini adalah klasifikasi 3 kelas kucing menggunakan CNN. Tiga kelas kucing yang digunakan adalah: 0-Domestic Cat, 1-Persian Cat, dan 2-Birman Cat. Data image untuk dataset sendiri selanjutnya dikumpulkan secara mandiri melalui internet. Agar penjelasan lebih mudah untuk disampaikan maka jumlah dataset yang digunakan tidak terlalu banyak, hanya 10 image untuk setiap kelasnya. Dataset yang digunakan sebagai contoh diberikan pada Gambar 30, 31, dan 32.



Gambar 30 Dataset kelas Kucing Domestik.

Gambar 31 Dataset kelas Kucing Persia.



Gambar 32 Dataset kelas Kucing Birman.

Menyeragamkan Ukuran Resolusi Image

Agar setiap data image dapat menjadi input bagi arsitektur CNN yang digunakan sebagai klasifier maka resolusi image perlu untuk diseragamkan. Pada tahapan ini akan dibuat kode program untuk me-*resize* image pada dataset.

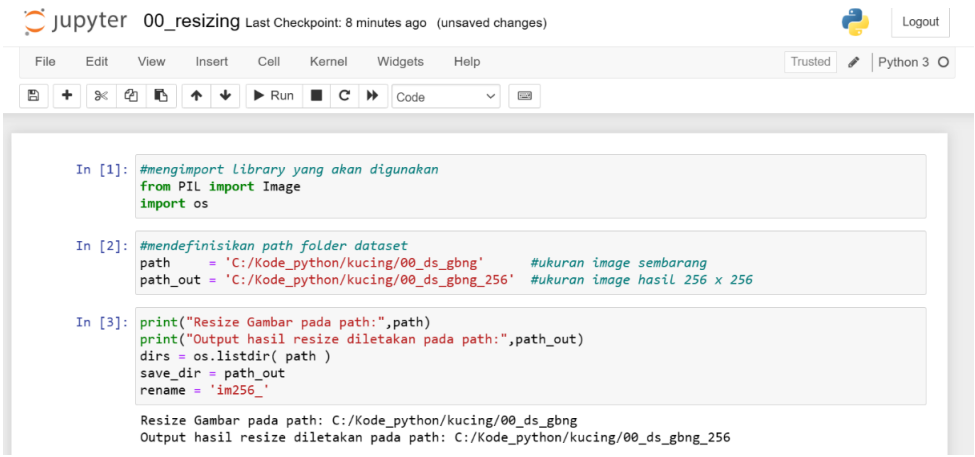
```
(base) C:\Kode_python\kucing>jupyter notebook
[I 15:10:12.001 NotebookApp] The port 8888 is already in use, trying another port.
[I 2023-01-16 15:10:12.657 LabApp] JupyterLab extension loaded from C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab
[I 2023-01-16 15:10:12.657 LabApp] JupyterLab application directory is C:\ProgramData\Anaconda3\share\jupyter\lab
[I 15:10:12.661 NotebookApp] Serving notebooks from local directory: C:\Kode_python\kucing
[I 15:10:12.661 NotebookApp] Jupyter Notebook 6.3.0 is running at:
[I 15:10:12.662 NotebookApp] http://localhost:8889/?token=1d16acb79806da0d45f25c9257cbb592d5ca1abeeb09698a
[I 15:10:12.662 NotebookApp]  or http://127.0.0.1:8889/?token=1d16acb79806da0d45f25c9257cbb592d5ca1abeeb09698a
[I 15:10:12.662 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 15:10:12.714 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/endan/AppData/Roaming/jupyter/runtime/nbserver-18304-open.html
    Or copy and paste one of these URLs:
        http://localhost:8889/?token=1d16acb79806da0d45f25c9257cbb592d5ca1abeeb09698a
     or http://127.0.0.1:8889/?token=1d16acb79806da0d45f25c9257cbb592d5ca1abeeb09698a
```

Gambar 33 Menjalankan Jupyter Notebook dari Anaconda Prompt.

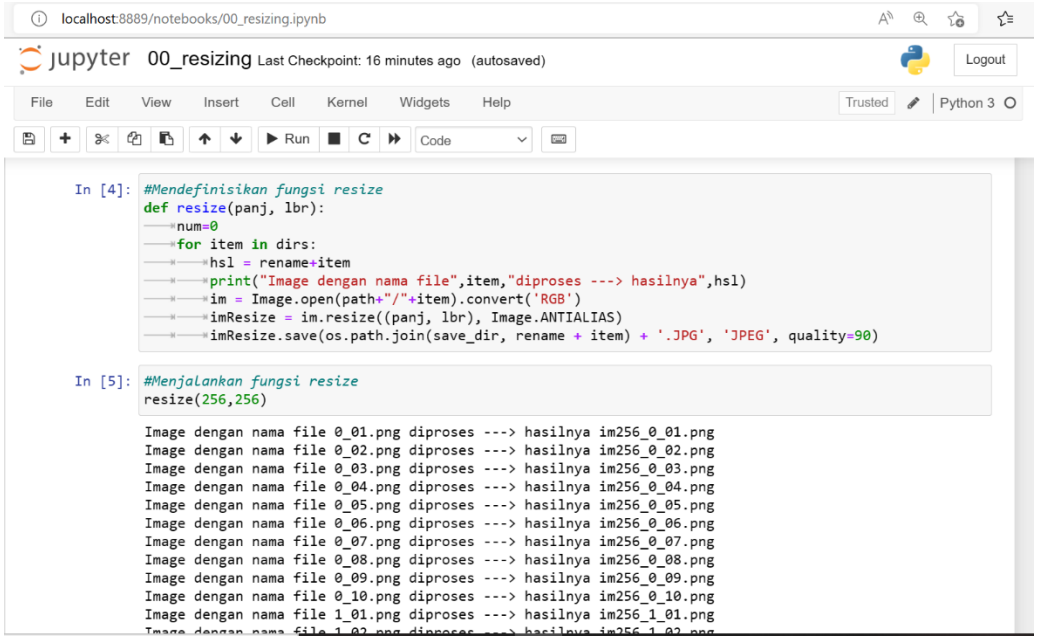
Langkah pertama yang kita lakukan adalah menjalankan Jupyter Notebook. Selain melalui Anaconda Navigator, Jupyter Notebook juga dapat di jalankan melalui Anaconda Prompt (gunakan perintah: jupyter notebook, Gambar 33). Sebagai hasilnya maka browser akan terbuka secara lokal seperti pada Gambar 34. Pada direktori aktif saat ini (C:/Kode_Python/kucing) terdapat tiga buah sub direktori yaitu:

- 00_ds folder tempat data image perkelas diletakkan pada masing-masing folder (masing-masing 10 image pada tiga folder kelas)
- 00_ds_gbng folder data seluruh image yang diletakan dalam satu folder (30 image), ukuran image masih sembarang
- 00_ds_gbng_256 merupakan folder kosong yang disediakan sebagai tempat menyimpan output hasil resize oleh program resizing.

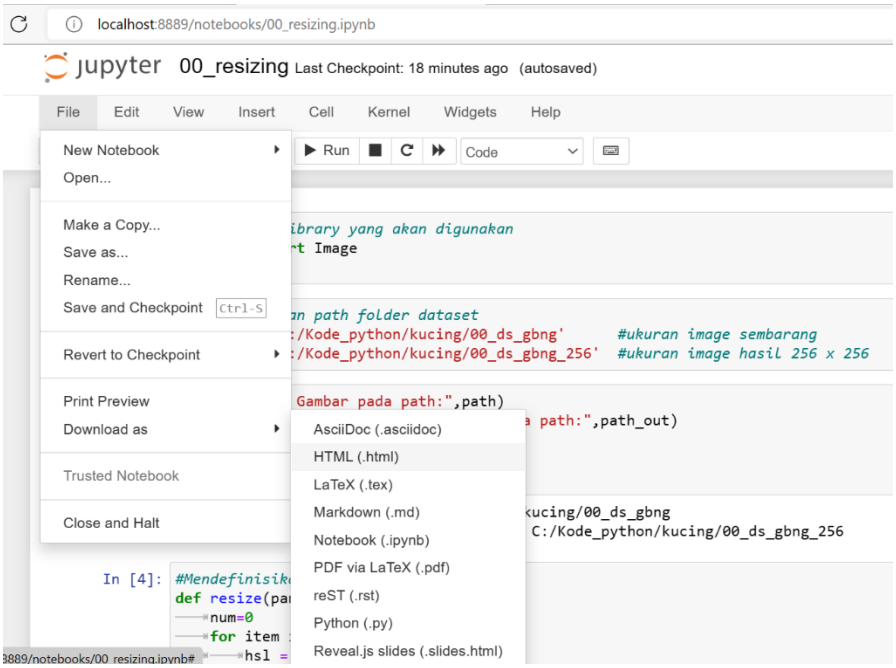
Gambar 34 Tampilan Jupyter Notebook pada path yang aktif.



Gambar 35 Library dan definisi variable untuk resize.



Gambar 36 Mendefinisikan dan menjalankan fungsi resize.



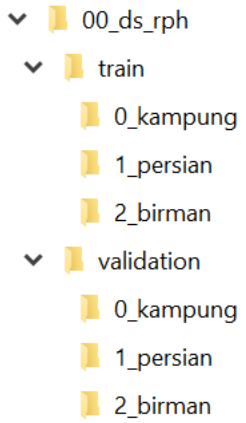
Gambar 37 Beragam format file download dari kode pada Jupyter Notebook.

Gambar 38 Format file kode pada Jupyter Notebook adalah ipynb.

Dengan membuka sebuah file kode program baru maka kita akan mulai mengetikkan kode program yang ditujukan untuk aksi resize image pada daaset. Di awali dengan mengimport package yang diperlukan dan mendeklarasikan variable (Gambar 35). Dilanjutkan dengan mendefinisikan sebuah fungsi bernama resize (Gambar 36) untuk kemudian menjalankan fungsi tersebut.

Hasil akhir dari penjalanan kode program ini akan menghasilkan 30 image hasil resize dari seluruh image yang ada di folder 00_ds_gbng. Adapun image hasil resize diletakan pada folder 00_ds_gbng_256. Seluruh image hasil resize diskenariokan berukuran 256x256 pixels. Sebagai informasi tambahan segala sesuatu kode program yang kita tuliskan pada Jupyter Notebook dapat kita simpan pada beragam format file yang tersedia (Gambar 37). Secara default kode program yang kia tuliskan pada Jupyter Noebook berekstensi file .ipynb (Gambar 38).

Skenario Eksperimen

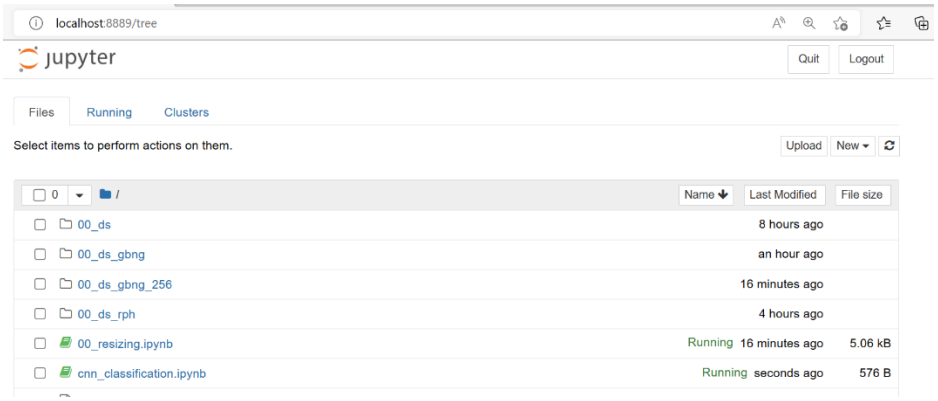


Gambar 39 Struktur direktori bagi dataset 00_ds_rph.

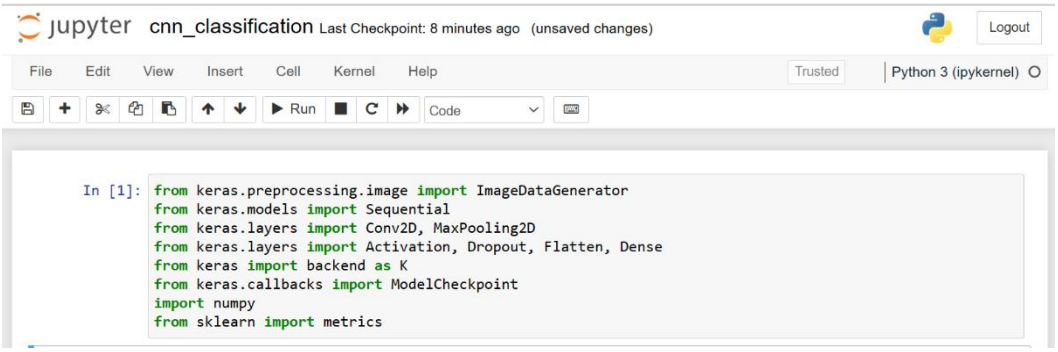
Dari seluruh image hasil resize pada folder 00_ds_gbng_256 maka masing-masing image akan kita letakkan sesuai dengan masing-masing folder kelasnya pada 00_ds_rph (Gambar 39). Sebagai skenario eksperimen pada kesempatan ini tujuh data image pada setiap kelas akan dijadikan sebagai data training, dan tiga data sisanya akan digunakan sebagai data validation. Secara total kita akan peroleh 21 data training dan hanya 9 buah data yang bertindak sebagai data validation.

Menggunakan CNN Sebagai Klasifier

Selanjutnya kita buat kode program baru pada Jupyter Notebook dan beri nama cnn_classification.ipynb (Gambar 40). Seperti pada umumnya kode kita awali dengan mendaftarkan seluruh package library yang akan digunakan (Gambar 41, pada backend kita juga membutuhkan PIL, jadi jangan lupa installkan pillow). Ada baiknya kita langsung run untuk memastikan apakah seluruh package sukses disertakan. Ketika ada pesan error maka kita harus menambahkan package yang gagal di-import tersebut. Setelah itu beragam variable yang digunakan akan ditentukan nilai-nilainya. Ukuran Image adalah 256x256 pixels, jumlah data training ada 21, jumlah data validasi ada 9, jumlah ulangan pada saat training ditentukan 200 putaran dengan batch pemrosesan pelatihan per-8 data (Gambar 42). Ukuran batch sebesar 8 akan menjadikan pada setiap putaran penuh ada 3 batch (kloter). Jika diperinci 8 data pada batch pertama, 8 data berikutnya pada batch kedua, dan 5 data sisanya pada batch ketiga.



Gambar 40 file kode cnn_classification.ipynb.



Gambar 41 Menyertakan beragam package yang akan digunakan.

Format data image sendiri ada dua jenis. Jenis yang pertama akan merepresentasikan bagian ukuran channelnya di depan. Pada jenis format data tersebut memiliki urutan parameter spesifikasi ukuran channel, lebar gambar, dan tinggi gambar. Adapun jenis format data image yang kedua akan meletakkan ukuran channel pada parameter ketiga. Agar kode program yang kita buat bisa memproses format data image yang keduanya, maka pembacaannya harus disesuaikan. Menggunakan sintaks kode seperti pada Gambar 43 maka tujuan tersebut dapat kita akomodasikan.

```
In [2]: img_width, img_height = 256, 256
        train_data_dir = '00_ds_rph/train'
        validation_data_dir = '00_ds_rph/validation'
        nb_train_samples = 21
        nb_validation_samples = 9
        epochs = 200
        batch_size = 8

        print("Ukuran image pada dataset:",img_width,"x",img_height)

        Ukuran image pada dataset: 256 x 256
```

Gambar 42 Mendefinisikan nilai-nilai variable yang digunakan.

```
In [3]: if K.image_data_format() == 'channels_first':
            input_shape = (3, img_width, img_height)
        else:
            input_shape = (img_width, img_height, 3)
```

Gambar 43 Memastikan pembacaan data sesuai format data input.

Gambar 44 Kode program untuk CNN.

```
In [4]: model = Sequential()
        model.add(Conv2D(32, (3, 3), input_shape=input_shape))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        model.add(Conv2D(64, (3, 3)))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        model.add(Flatten())
        model.add(Dense(64))
        model.add(Activation('relu'))
        model.add(Dropout(0.5))
        model.add(Dense(3))
        model.add(Activation('softmax')) #untuk multiclass nih kan 3 kelas

        model.summary()
```

Gambar 45 Kode program untuk Arsitektur yang digunakan CNN.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 activation (Activation)     (None, 254, 254, 32)      0

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 activation_1 (Activation)   (None, 125, 125, 64)      0

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 flatten (Flatten)           (None, 246016)            0

 dense (Dense)               (None, 64)                15745088

 activation_2 (Activation)   (None, 64)                0

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 3)                 195

 activation_3 (Activation)   (None, 3)                 0

=================================================================
Total params: 15,764,675
Trainable params: 15,764,675
Non-trainable params: 0
_________________________________________________________________
```

Gambar 46 Layer Arsitektur CNN yang digunakan.

Perlu ditekankan bahwa nilai kinerja model yang dibuat belum ditujukan untuk menghasilkan model yang akurat namun lebih kepada hanya sebagai ilustrasi mulai dari rancangan logik (Gambar 44) hingga implementasinya menggunakan package library Keras seperti pada Gambar 45. Arsitektur CNN yang digunakan terdiri atas sebuah layer input, dua pasang *convolution* dan *pooling*, sebuah *fully connected layer* (dense), dan sebuah layer output. Secara detail jumlah parameter yang ditrain pada setiap layer dapat kita tampilkan dengan memanggil method summary dari object model yang kita gunakan (Gambar 46). Pada setiap pasangan layer convolution dan pooling diberikan fungsi aktifasi ReLu dan pada layer output untuk memperoleh agregasi penilaian digunakan fungsi softmax.

kali ini merupakan kategorik class dengan kelas sebanyak tiga. Berdasarkan kondisi tersebut maka pastikan mode kelas dan teknik penghintungan loss juga harus categoric (Gambar 47).

```
In [5]: model.compile(loss="categorical_crossentropy", optimizer= "adam", metrics=['accuracy'])

In [6]: # Augmentation configuration untuk tahapan training
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# Augmentation configuration yang digunakan untuk testing hanya rescalling
test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical',
    #pastikan nama kelas sama dengan nama direktori
    classes = ["0_kampung", "1_persian" , "2_birman"])

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical',
    classes = ["0_kampung", "1_persian" , "2_birman"],
    shuffle = False)

Found 21 images belonging to 3 classes.
Found 9 images belonging to 3 classes.
```

Gambar 47 Setting tahapan pelatihan dan penerapan augmentasi data.

Pada ilustrasi di kesempatan ini diterapkan juga augmentasi data. Hal ini ditujukan untuk meningkatkan variasi informasi pada image. Bentuk transformasi pada data training untuk augmentasi meliputi pensklaan, shear, zoom range, dan horizontal flip. Adapun untuk data testing hanya diterapkan teknik rescalling. Pada penerapannya digunakan objek dari kelas image_generator.

Hasil Eksperimen

Proses pelatihan untuk pembentukan model menggunakan method fit(). Karena model CNN memiliki aspek stokastik terkait titik konvergensi model yang tidak selalu dicapai pada epoch terakhir, tentu perlu prosedur yang dapat menangkap model terbaik pada sembarang iterasi. Untuk mencapai tujuan ini maka digunakan fungsi ModelCheckpoint (Gambar 48). Output tahapan training dari model klasifikasi yang dibentuk pada Gambar 49. Seperti terlihat pada Gambar 49 nilai akurasi dan loss masih bersifat fluktuatif pada *epoch* awal.

```
In [7]: mcp_save = ModelCheckpoint('model_terbaik.hdf5',
                              save_best_only=True, monitor='val_loss', mode='min')

In [8]: model.fit(train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=epochs,callbacks=[mcp_save],
    validation_data=validation_generator,
    validation_steps=nb_validation_samples // batch_size)
```

Gambar 48 Setting tahapan pelatihan dan penerapan augmentasi data.

```
Epoch 1/200
2/2 [==============================] - 3s 2s/step - loss: 3.5409 - accuracy: 0.1875 - val_loss: 2.4105 - val_accuracy: 0.3750
Epoch 2/200
2/2 [==============================] - 1s 723ms/step - loss: 3.8745 - accuracy: 0.1875 - val_loss: 1.1494 - val_accuracy: 0.2500
Epoch 3/200
2/2 [==============================] - 1s 1s/step - loss: 1.8590 - accuracy: 0.2500 - val_loss: 1.1133 - val_accuracy: 0.2500
Epoch 4/200
2/2 [==============================] - 1s 1s/step - loss: 0.8981 - accuracy: 0.6250 - val_loss: 1.0326 - val_accuracy: 0.3750
Epoch 5/200
2/2 [==============================] - 0s 287ms/step - loss: 1.1627 - accuracy: 0.2308 - val_loss: 1.0732 - val_accuracy: 0.6250
Epoch 6/200
2/2 [==============================] - 1s 274ms/step - loss: 1.0055 - accuracy: 0.4375 - val_loss: 1.0575 - val_accuracy: 0.6250
Epoch 7/200
2/2 [==============================] - 1s 1s/step - loss: 1.0524 - accuracy: 0.5385 - val_loss: 1.0185 - val_accuracy: 0.7500
Epoch 8/200
2/2 [==============================] - 1s 1s/step - loss: 0.8829 - accuracy: 0.7692 - val_loss: 0.9954 - val_accuracy: 0.5000
                                                    ...
Epoch 198/200
2/2 [==============================] - 1s 337ms/step - loss: 6.2010e-04 - accuracy: 1.0000 - val_loss: 1.8796 - val_accuracy: 0.7500
Epoch 199/200
2/2 [==============================] - 1s 284ms/step - loss: 4.5316e-04 - accuracy: 1.0000 - val_loss: 1.8846 - val_accuracy: 0.7500
Epoch 200/200
2/2 [==============================] - 1s 207ms/step - loss: 0.0060 - accuracy: 1.0000 - val_loss: 1.9331 - val_accuracy: 0.7500
<keras.callbacks.History at 0x22532cfee20>
```

Gambar 49 Proses tahapan training.

```
In [9]: pred=model.predict(validation_generator,verbose=1,steps=nb_validation_samples/batch_size)
        1/1 [==============================] - 0s 104ms/step

In [10]: y_act  = validation_generator.classes

In [11]: print("Skor Prediksi:\n",pred)
         Skor Prediksi:
          [[1.4953190e-04 9.9985003e-01 3.0658771e-07]
          [9.7448218e-01 2.5512150e-02 5.5928185e-06]
          [9.9999994e-01 3.1542860e-14 4.5467587e-17]
          [4.0006771e-06 9.9956876e-01 4.2722677e-04]
          [9.9861327e-06 9.9998909e-01 8.8403277e-07]
          [4.4045206e-03 1.3197399e-03 9.9427569e-01]
          [2.4299872e-30 9.9667646e-21 9.9999994e-01]
          [2.5745389e-16 1.1021402e-11 9.9999994e-01]
          [2.9200094e-22 6.2823749e-14 1.0000000e+00]]
```

Gambar 50 Prediksi skor terhadap data validation.

Berdasarkan hasil training maka diperoleh model dengan tingkat akurasi sebesar 75% (Gambar 49). Prediksi skor secara detail dari ke sembilan data testing diberikan pada Gambar 50. Untuk memperoleh perbandingan antara kelas hasil prediksi dengan kelas actual sebagai *groundtruth* diberikan pada Gambar 51. Pada kesempatan ini kita membuat sebuah fungsi kelas_pred untuk mengkonversi dari nilai skor prediksi kepada kelas kategori spesifik. Kelas kategori prediksi adalah kolom dengan nilai skor prediksi. Untuk mendapatkan representasi yang lebih lengkap maka dibuat confusion matrix beserta beragam parameter evaluasi yang dari model Gambar 52.

```
In [12]: def kelas_pred(P):
             n = len(P)
             CL = []
             #print(n)
             for i in range(n):
                 c = numpy.where(P[i] == numpy.amax(P[i]))
                 CL.append(c[0][0])
                 #print('c0:',c[0][0])
             return CL

         y_pred = kelas_pred(numpy.array(pred))
         print("\n\nKelas Prediksi:",y_pred)
         print("Kelas Groundtruth:",y_act)
```

```
Kelas Prediksi: [1, 0, 0, 1, 1, 2, 2, 2, 2]
Kelas Groundtruth: [0 0 0 1 1 1 2 2 2]
```

Gambar 51 Kelas prediksi versus kelas aktual.

```
In [13]: cm = metrics.confusion_matrix(y_act, y_pred, labels=[0,1,2])
         print("\nDETAIL:\n",metrics.classification_report(y_act, y_pred, labels=[0,1,2]))
         print("\nCONFUSION MATRICES:\n",cm)
```

```
DETAIL:
              precision    recall  f1-score   support

           0       1.00      0.67      0.80         3
           1       0.67      0.67      0.67         3
           2       0.75      1.00      0.86         3

    accuracy                           0.78         9
   macro avg       0.81      0.78      0.77         9
weighted avg       0.81      0.78      0.77         9


CONFUSION MATRICES:
 [[2 1 0]
 [0 2 1]
 [0 0 3]]
```

Gambar 52 *Metrics* hasil evaluasi dan *confusion matrices* dari model.

Diskusi dan Pembahasan

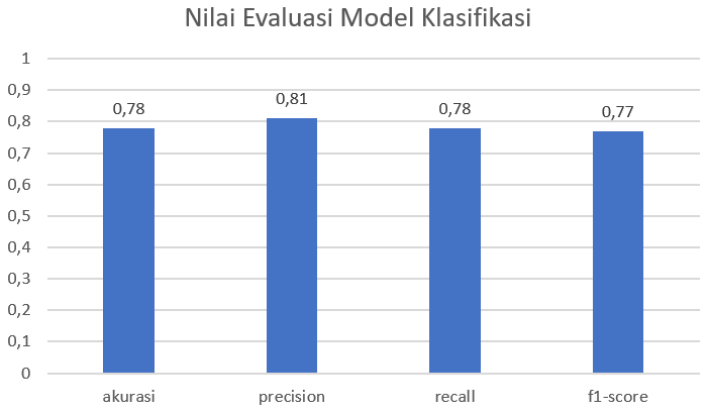
Penomoran Id kelas secara berturut-turut adalah 0-domestic cat, 1-persian cat, dan 2-birman cat. Mengacu pada Tabel 1, dari sembilan data validasi berhasil diprediksi dengan tepat sebanyak tujuh data. Khusus untuk data kelas birman berhasil diprediksi 100% dengan tepat. Adapun masing-masing untuk kelas domestic cat dan persian cat masing-masing keliru memprediksi pada satu data validasi.

Kesalahan prediksi pada kelas domestic cat terjadi pada data im256_0_08.png. Pada kasus ini domestic cat diprediksi oleh model sebagai persian cat. Hal ini diperkirakan dapat terjadi karena warna dominan pada data yang diprediksi untuk warna abu-abu memiliki jarak pengambilan citra dari data validasi yang agak berbeda dengan data training untuk kelas domestic cat.

Tabel 1. Nilai pemetaan prediksi setiap data validasi yang digunakan

Image	 im256_0_08.png	 im256_0_09.png	 im256_0_10.png	 im256_1_08.png	 im256_1_09.png	 im256_1_10.png	 im256_2_08.png	 im256_2_09.png	 im256_2_10.png
Kelas Aktual	0	0	0	1	1	1	2	2	2
Kelas prediksi	1	0	0	1	1	2	2	2	2

Selanjutnya untuk kelas persian cat, kesalahan prediksi terjadi pada data im256_1_10. Pada kasus ini data persian cat diprediksi sebagai birman cat. Jika kita perbandingkan antara data yang salah diprediksi ini dengan data training dari kelas persian terlihat bahwa orientasi data validasi agak miring. Hasil ini mengindikasikan bahwa teknik augmentasi pada data training belum cukup memadai untuk menjadikan model dapat memprediksi secara akurat terhadap data yang memiliki orientasi berbeda terhadap data trainingnya.



Gambar 53 *Metrics* hasil evaluasi dan *confusion matrices* dari model.

Nilai akurasi, precision, recall, dan F1score dari studi kasus ini diberikan pada Gambar 53. Beberapa rekomendasi utama yang dapat ditempuh untuk meningkatkan kinerja dari model klasifikasi di antaranya memperbanyak data dan variasi pada data training, memperdalam arsitektur CNN yang digunakan, dan memperbanyak jumlah epoch serta teknik augmentasi yang digunakan.

4 KESIMPULAN

Seperti telah disebutkan sebelumnya bahwa pada kesempatan ini model yang dibentuk belum ditujukan untuk memperoleh model klasifikasi yang akurat. Pada studi kasus penelitian yang disampaikan ukuran dataset masih sangat sedikit, secara total hanya 30 data dengan tiga kelas kategori. Meskipun demikian sesuai dengan tujuan utamanya, dokumen ini diharapkan dapat menjadi referensi awal bagi pelaku penelitian yang menggunakan python sebagai bahasa pemrogramannya. Pada dokumen ini memaparkan secara lengkap mulai dari tahapan persiapan lingkungan perangkat pendukung pemrograman python, sampai dengan contoh pemanggilan beragam fungsi yang sering digunakan pada eksperimen penelitian melalui pemberian kasus penelitian sederhana sebagai ilustrasi.

DAFTAR PUSTAKA

Anaconda.com - https://docs.anaconda.com/anaconda/install/windows/

Ciresan CD, et al. 2011. Flexible, High Performance Convolutional Neural Networks for Image Classification. Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI'11), Vol. Two, Pages 1237-1242.

Giri EP, *et al.* 2016. Ischemic Stroke Identification Based on EEG and 1D Convolutional Neural Network. Proceeding International Conference on Advanced Komputer Science and Information Systems (ICACSIS) 2016.

Hinton GE & Salakhutdinov RR. 2006. Reducing the Dimensionality of Data with Neural Networks. Science 28 Jul 2006: Vol. 313, Issue 5786, pp. 504-507 DOI: 10.1126/science.1127647.

Hamid A S, Mohamed A, Jiang H, & Pen G. 2012. Applying Convolutional Neural Networks Concepts to Hybrid NN-HMM Model for Speech Recognition. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 4277-4280 25-30 March 2012, Kyoto, Japan.

Hamid AS, Mohamed AR, Jiang H, Deng L, Penn G & Yu D. 2014. Convolutional Neural Networks for Speech Recognition. IEEE/ACM Transactions On Audio, Speech, And Language Processing, VOL. 22, NO. 10, October 2014.

Lecun Y, Bottou Y, Bengio Y and Haffner P. 1998. "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

Li D. 2014. Overview paper: A tutorial survey of architectures, algorithms, and applications for deep learning. SIP (2014), vol. 3, e2, page 1 of 2http://creativecommons.org/licenses/by/3.0/, doi:10.1017/ATSIP.2013.99.

Repository Anaconda - https://repo.anaconda.com/archive/Anaconda3-2022.10-Windows-x86_64.exe

Sultana F, Sufian A, Dutta P. 2018. Advancements in Image Classification using Convolutional Neural Network. 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), Kolkata, India, 2018, pp. 122-129, doi: 10.1109/ICRCICN.2018.8718718.

Zheng Y, Liu Q, Chen E, Ge Y & Zhao JL. 2014. Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks. Lecture Notes in Komputer Science, Volume 8485, pp 298-310.