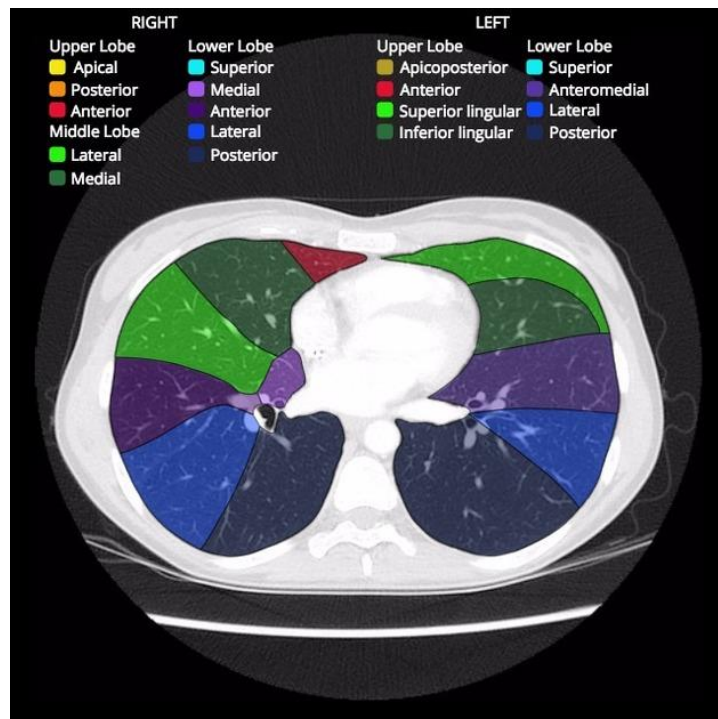


BAB 2 LANDASAN TEORI

2.1. Citra CT Toraks

Computed Tomography Scan (CT Scan) adalah teknik pencitraan digital dengan menggabungkan banyak citra X-ray menggunakan tabung yang berputar, sehingga dari hasil pemindaian tersebut dapat dihasilkan potongan-potongan citra penampang paru-paru. Sedangkan toraks mengacu pada bagian rongga dada. Maka citra CT pada bagian toraks dapat digunakan untuk mendiagnosis penyakit atau kelainan pada rongga dada yang meliputi bagian paru-paru, termasuk pada bagian vaskuler atau yang berhubungan dengan pembuluh darah pada paru-paru, serta pada bagian mediastinum.

Diagnosis pada citra CT toraks tidak terlepas dari segmentasi paru-paru untuk mengacu lokasi terjadinya kelainan atau penyakit yang teridentifikasi. Segmentasi paru-paru pada citra CT toraks dapat dilihat pada **Gambar 2.1** [11].



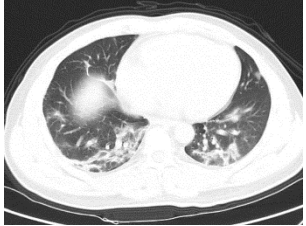
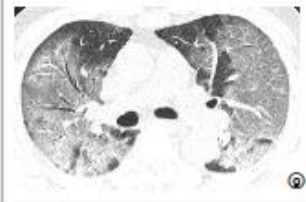
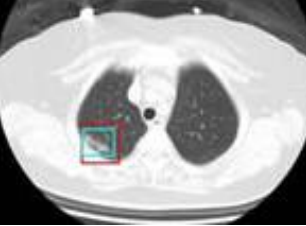
Gambar 2.1 - Segmentasi paru-paru pada citra CT toraks


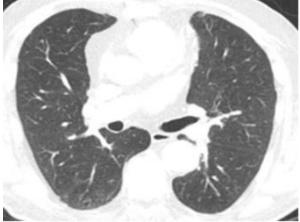
2.1.1. Ciri COVID-19 pada Citra CT Toraks

Kelainan-kelainan yang terdeteksi pada citra CT pasien COVID-19 bersifat variatif. Tanda umum yang kerap muncul antara lain adanya konsolidasi, *Ground Glass Opacities* (GGO), nodul, bronchogram udara, penebalan-penebalan pada dinding paru, dsb.

Pada konsolidasi dan GGO, terdapat beberapa pola penyebaran atau bentuk yang dapat diamati, diantaranya penyebaran-penyebaran yang berbentuk atau bersifat *focal, diffuse, patchy, multifocal, nodular, centrilobular, mosaic, crazy paving, halo sign* dan *reversed halo sign* [12]. Dan menurut tempat terjadinya, temuan-temuan yang telah disebutkan dapat terjadi pada beberapa lokasi secara *bilateral, subpleural, peripheral, interlobular, intralobular, centrilobular*, dsb [13]. Beberapa sampel citra CT toraks yang digunakan pada penelitian ini lengkap dengan deskripsinya dapat dilihat pada **Tabel 2.1**.

Tabel 2.1 - Sampel data CT Scan

Citra	Kelas	Deskripsi Citra
	ct_covid	<i>bilateral focal consolidation, lobar consolidation, and patchy consolidation were clearly observed, especially in the lower lung</i>
	ct_covid	<i>patchy GGO distributed in subpleural area in left lower lung</i>
	ct_covid	<i>ground glass nodular opacities on admission and progressed to multiple patchy ground glass opacities on reexamination. peripherally distributed focal ground glass nodular opacities in only right lung</i>

Citra	Kelas	Deskripsi Citra
	ct_noncovid	<i>healthy lung</i>
	ct_noncovid	<i>healthy lung</i>

2.2. Citra Digital

Citra digital adalah citra yang terdiri dari elemen gambar yang disebut pixel. Setiap pixel pada citra digital direpresentasikan oleh tepat satu buah warna. Sehingga sebuah citra digital dapat dikatakan secara konsep adalah kumpulan piksel berwarna dengan jumlah banyak yang dipadu untuk membentuk sebuah citra. Representasi warna pada setiap pixel pada citra digital tersimpan dalam bentuk angka atau digit, dari situlah penyebutan digital berasal. Citra digital dipetakan terhadap sumbu x dan y pada sumbu koordinat spasial. Sehingga apa yang disebut citra digital akan memiliki ukuran pixel dalam bentuk dua dimensi atau dikenal dengan resolusi, ditambah sebuah dimensi dengan kedalaman kanal-kanal warna.

2.2.1. Citra *Grayscale*

Citra *grayscale* adalah citra yang pada setiap pikselnya terdapat hanya satu nilai kanal. citra *grayscale* menyimpan nilai dalam format 8 bit untuk setiap pikselnya, sehingga dimungkinkan terdapatnya 256 nilai keabuan yang berbeda. Pada citra *grayscale*, keabuan dengan tingkat kecerahan yang paling rendah bernilai 0 berupa warna hitam, dan paling cerah bernilai 255 berupa warna putih [14].

2.2.2. Citra RGB

Citra RGB adalah citra yang memiliki tiga nilai kanal pada setiap pikselnya yaitu nilai *red*, *green*, dan *blue*. Setiap kanal pada RGB masing-masing memiliki nilai hingga 256 intensitas, yaitu 0 hingga 255 [14].

2.3. Pengolahan Citra Digital

Pengolahan citra digital merujuk pada pemrosesan citra yang dilakukan dengan menggunakan komputer. Pengolahan yang dilakukan adalah dengan memproses dan mengubah nilai-nilai piksel pada data citra yang digunakan.

2.3.1. *Resize*

Resizing adalah proses pengubahan ukuran citra, baik itu memperbesar atau memperkecilnya. Terdapat berbagai macam algoritma yang dapat digunakan untuk keperluan tersebut, diantaranya dengan *nearest-neighbor interpolation*, *bilinear interpolation*, *bicubic interpolation*, *sinc resampling*, *lanczos resampling*, *box sampling*, *mipmap*, *fourier transform*, *edge-directed interpolation*, dsb.

2.3.2. Konversi Warna

Konversi warna yang dimaksud adalah proses pengubahan dimensi kedalaman warna. Termasuk diantaranya pengubahan citra RGB menjadi *grayscale* dan juga sebaliknya. Untuk mengubah citra RGB menjadi *grayscale*, dapat digunakan rumus untuk mendapatkan nilai piksel keabuan seperti berikut:

$$Gray = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad (2.1)$$

Sedangkan untuk mengubah citra *grayscale* menjadi RGB dapat dilakukan dengan cara memberikan nilai *Red*, *Green*, dan *Blue* dengan nilai yang sama. Ketiga kanal tersebut akan memiliki nilai sama berdasarkan nilai keabuan dari citra *grayscale* yang digunakan. Dalam hal ini, nilai *gray* sebuah piksel dijadikan acuan pada piksel tersebut untuk ketiga kanal RGB yang akan dihasilkan, sehingga nantinya didapatkan piksel dalam tiga kanal dengan nilai $R = G = B$.

2.4. Ekstraksi Fitur Citra

Ekstraksi fitur pada citra adalah proses untuk mendapatkan fitur atau ciri utama dari sebuah citra. Ekstraksi fitur merupakan tahapan penting dalam penelitian ini karena klasifikasi tidak dapat dilakukan tanpa vektor fitur yang dihasilkan dengan proses ekstraksi tersebut. Pada penelitian ini, vektor fitur didapat dengan melakukan ekstraksi fitur menggunakan sebuah model CNN bernama VGG-16.

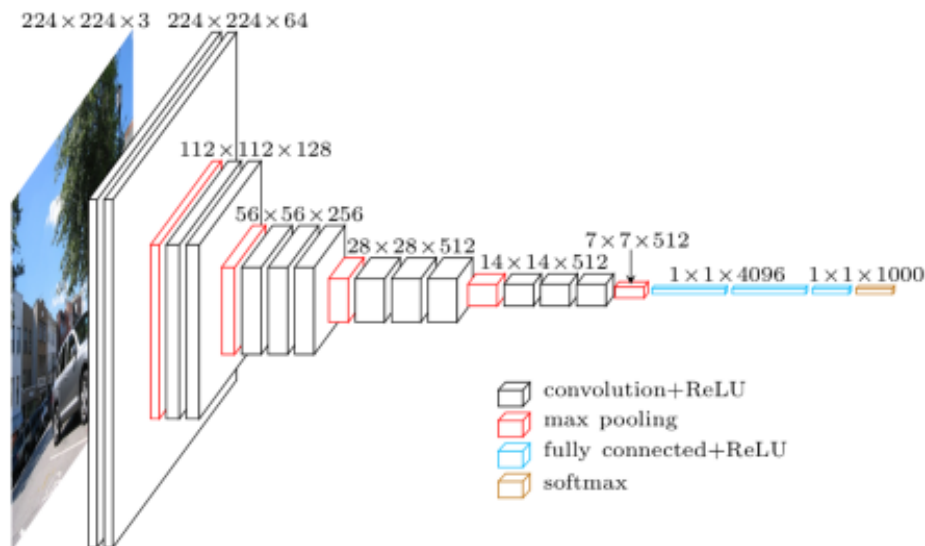
2.5. VGG-16: Arsitektur *Convolutional Neural Network* untuk Citra

Convolutional Neural Network (CNN) secara garis besar adalah jaringan saraf tiruan yang umum digunakan untuk citra. Kelebihan CNN dibanding algoritma lain untuk ekstraksi fitur citra adalah pembelajaran yang dilakukan secara otomatis dengan *preprocessing* minimal karena lebih difokuskan untuk memanfaatkan jaringan *neuron* yang ada tanpa perlu pengetahuan tambahan mengenai citra yang akan dikenali. Secara umum terdiri dari jaringan *convolution* yang saling berhubungan dan masing-masing melakukan operasi konvolusi dengan masukan dua buah fungsi untuk menghasilkan fungsi baru. Pada CNN juga terdapat fungsi aktivasi yang memutuskan pengaktifkan tiap *neuron*.

Adapun VGG-16 adalah sebuah arsitektur CNN dengan konfigurasi tertentu yang dirancang oleh K. Simonyan dan A. Zisserman dari University of Oxford, pada *paper* mereka yang berjudul “*Very Deep Convolutional Networks for Large-Scale Image Recognition*”. Model tersebut mendapat peringkat 5 teratas pada ILSVRC tahun 2014, yang merupakan sebuah *challenge* untuk pengenalan citra berskala besar pada dataset ImageNet yang berisikan lebih dari 14 juta data citra dengan 1000 kelas yang berbeda. Model tersebut menghasilkan akurasi pengujian sebesar 92.7% [15].

VGG-16 sebagai CNN memiliki 13 *convolutional layer* dan 3 *fully connected layer*, sehingga totalnya adalah 16. VGG-16 memiliki Ukuran filter konvolusi yang kecil, yaitu 3×3 . Citra masukan untuk lapisan konvolusi pertama berdimensi seragam, yaitu $224 \times 224 \times 3$ atau citra RGB yang berukuran 224×224 . Ukuran *stride* pada lapisan konvolusi memiliki ketentuan ukuran yaitu 1 pixel. Adapun ukuran *stride* pada *max-pooling* adalah 2 dengan ukuran filter 2×2 . Dan

lapisan-lapisan yang ada juga dilengkapi dengan *rectifier* atau ReLU sebagai fungsi aktivasinya. Namun khusus pada lapisan terakhir dilengkapi dengan fungsi aktivasi *softmax*. Untuk lebih jelasnya, gambaran arsitektur VGG-16 dapat dilihat pada **Gambar 2.2** [16].



Gambar 2.2 - Arsitektur VGG-16

2.6. Machine Learning

Machine Learning adalah metode analisis data yang mengotomasi pembangunan model pembelajaran.

2.6.1. Klasifikasi

Klasifikasi merupakan suatu proses pembelajaran mesin yang melakukan penilaian terhadap suatu obyek data untuk masuk ke dalam suatu kelas tertentu dari sejumlah kelas yang tersedia. Terdapat 2 proses utama pada klasifikasi, yaitu pelatihan (*Training*) dan pengujian (*Testing*).

2.7. Support Vector Machine (SVM)

Support Vector Machine (SVM) adalah suatu model *supervised learning* yang dikembangkan oleh Vladimir Vapnik, Bernhard Boser, dan Isabelle Guyon antara tahun 1992 hingga 1997 [17]. SVM bekerja dengan mencari sebuah *hyperplane* paling optimal yang memisahkan setidaknya dua set data dari kelas yang berbeda. Jika dilihat dari konsep geometri dan juga sebaran data, maka *hyperplane* adalah sebuah bidang yang memiliki kedalaman 1 dimensi lebih rendah dibanding dimensi ruang dimana titik-titik data tersebar. *Hyperplane* diposisikan sedemikian rupa sehingga letak *hyperplane* terhadap masing-masing data terdekat pada setiap kelas dapat memiliki margin atau jarak yang optimal untuk memisahkan setiap kelas. Adapun masing-masing titik data yang paling dekat dengan *hyperplane* dari setiap kelas inilah yang disebut *support vector* [18].

Pada kasus *machine learning*, SVM dapat digunakan pada berbagai jenis data masukan. Misalkan untuk kasus klasifikasi, SVM kerap digunakan untuk data berbentuk teks atau data berbentuk citra. Pada kasus data teks, salah satu contohnya adalah klasifikasi dokumen. Sebuah penelitian tentang klasifikasi dokumen menggunakan peringkasan menghasilkan akurasi sebesar 78% untuk dokumen tanpa peringkasan dan 77% untuk dokumen dengan peringkasan menggunakan SVM [19]. Pada penelitian lain dengan masukan data citra, SVM digunakan untuk pengenalan ekspresi wajah. Hasilnya model pada penelitian mendapatkan akurasi 97,41% dengan menggunakan Markov Stationary Feature - Vector Quantization (MSF-VQ) sebagai metode ekstraksi fiturnya [20].

Secara prinsip, SVM adalah sebuah *linear classifier* yang merupakan metode untuk memisahkan 2 jenis kelas yang berbeda. Setiap data dinyatakan dalam x_i dan y_i . dimana x_i adalah vektor fitur pada sebuah data, dan y_i menyatakan kelas atau kategori data tersebut. Adapun jenis kelas dapat dinyatakan dengan dengan label bernilai -1 dan $+1$. Klasifikasi data dilakukan dengan melakukan:

$$\text{sign}(f(x)) \tag{2.2}$$

Dimana $f(x)$ adalah fungsi untuk mencari nilai *hyperplane*. Hasil dari persamaan 2.2 tersebut adalah sebuah nilai label. Apabila nilai $f(x) < 0$, maka data dilabeli negatif atau $y_i = -1$. Dan apabila nilai $f(x) > 0$, maka data dilabeli positif atau $y_i = +1$. Adapun nilai $f(x)$ didapatkan dengan menggunakan persamaan berikut:

$$f(x) = \sum_{i=1}^N w \cdot x_i + b \quad (2.3)$$

Dimana:

x_i = Vektor Fitur data ke-i

y_i = Label data ke-i

w = Nilai dari bidang normal

b = Posisi bidang relatif terhadap pusat koordinat

2.7.1. Kernel Trick

Pada umumnya, masalah dalam *domain* dunia nyata jarang yang bersifat *linear separable*. Maka untuk menyelesaikan masalah-masalah non linear, model SVM dapat dimodifikasi dengan menggunakan fungsi *Kernel*. Terdapat beberapa macam *kernel* yang dapat digunakan dalam SVM selain linear, diantaranya *kernel* polynomial, Radial Basic Function (RBF), dan sigmoid. Perbedaan fungsi dari setiap *kernel* dapat dilihat pada Tabel 2.2.

Tabel 2.2 Jenis Kernel dengan Fungsinya

<i>Kernel</i>	Fungsi <i>Kernel</i>
<i>Linear</i>	$K(x, x_i) = x \cdot x_i$
<i>Polynomial</i>	$K(x, x_i) = [\gamma \cdot (x \cdot x_i) + coef]^d$
<i>RBF</i>	$K(x, x_i) = \exp(-\gamma \cdot \ x - x_i\ ^2)$
<i>Sigmoid</i>	$K(x, x_i) = \tanh(\gamma(x \cdot x_i) + coef)$

2.8. Pengukuran Performa Klasifikasi

Pengukuran performa klasifikasi bertujuan untuk mendapatkan nilai yang representatif terhadap kemampuan suatu *classifier* melakukan tugasnya memprediksi kelas suatu data.

2.8.1. Confusion Matrix

Metrik ukur bisa didapatkan dengan menghitung dan menempatkan perbandingan kebenaran prediksi pada tabel *confusion matrix*, yaitu sebuah tabel yang menyatakan jumlah data uji yang benar diklasifikasikan dan jumlah data uji yang salah diklasifikasikan [21]. Contoh *confusion matrix* untuk klasifikasi biner ditunjukkan pada **Tabel 2.3**.

Tabel 2.3 - Tabel Confusion Matrix

		Kelas Prediksi	
		+1	-1
Kelas Sebenarnya	+1	TP	FN
	-1	FP	TN

Keterangan untuk tabel dinyatakan sebagai berikut:

1. *True Positive* (TP): yaitu jumlah dokumen dari kelas 1 yang benar dan diklasifikasikan sebagai kelas 1.
2. *True Negative* (TN): yaitu jumlah dokumen dari kelas 0 yang benar diklasifikasikan sebagai kelas 0.
3. *False Positive* (FP): yaitu jumlah dokumen dari kelas 0 yang salah diklasifikasikan sebagai kelas 1.
4. *False Negative* (FN): yaitu jumlah dokumen dari kelas 1 yang salah diklasifikasikan sebagai kelas 0.

2.8.2. Metrik Akurasi

Dari *confusion Matrix*, dapat diekstrak nilai-nilai metrik pengukuran, diantaranya accuracy, precision, recall (sensitivity), F1-score.

Metrik Akurasi adalah ukuran jumlah semua prediksi yang benar dibagi oleh total jumlah dataset, dimana nilai terbaiknya adalah 1 dan terburuknya adalah 0. Adapun rumus akurasi *confusion matrix* adalah sebagai berikut:

$$Akurasi = \frac{TP + TN}{TP + FP + FN + TN}$$

2.9. Python

Python adalah bahasa pemrograman interpretatif yang dianggap mudah dipelajari. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, Python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami syntax. Dengan kata lain, Python diklaim sebagai bahasa pemrograman yang memiliki kode-kode pemrograman yang sangat jelas, lengkap, dan mudah untuk dipahami. Python secara umum berbentuk pemrograman berorientasi objek, imperatif, dan fungsional.

Python dikembangkan oleh Guido van Rossum pada tahun 1990 di CWI, Amsterdam sebagai kelanjutan dari bahasa pemrograman ABC. Sampai saat ini Python masih dikembangkan oleh Python Software Foundation. Bahasa Python mendukung hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distronya sudah menyertakan Python di dalamnya. Dengan kode yang simpel dan mudah diimplementasikan, seorang pemrograman dapat lebih mengutamakan pengembangan aplikasi yang dibuat, bukan malah sibuk mencari syntax error [20].

2.9.1. Pandas

Pandas merupakan *library* Python yang berguna untuk manipulasi dan analisis data. Pandas terutama digunakan untuk operasi-operasi pada tabel angka. Nama pandas diturunkan dari “panel data” yang merupakan istilah pada statistika berkenaan data multi-dimensi yang diukur terhadap berjalannya waktu. Selain itu nama Pandas juga juga kepanjangan dari frasa “Python data analysis”.

Pandas mulai dikembangkan oleh Wes McKinney saat dia menjadi peneliti di AQR Capital pada tahun 2008, untuk memenuhi kebutuhan akan sebuah *tool* yang berperforma tinggi dapat melakukan analisis kuantitatif pada data finansial.

Pada tahun 2012, Chang She membantu mengembangkan *library* Pandas. Lalu pada tahun 2015, Pandas mendaftarkan diri sebagai proyek yang disponsori oleh lembaga amal non-profit NumFocus di Amerika Serikat.

Fitur-fitur utama pada pandas antara lain *dataframe*, *tools* untuk membaca dan menulis data antar memori, struktur data, dan format file berbeda, *reshaping*, *slicing*, *indexing*, *insertion*, *deletion*, *merging*, *joining*, *split*, dsb. Pandas ditulis dalam bahasa Python, Cython, dan C [21].

2.9.2. Scikit-Learn

Scikit-learn adalah *library* Python yang menangani berbagai macam algoritma klasifikasi, regresi dan *clustering*. Termasuk di dalamnya SVM, *random forest*, *gradient boosting*, dsb. *Library* ini juga didesain agar dapat saling beroperasi dengan *library* Numpy dan SciPy. Scikit-learn ditulis dalam bahasa Python, Cython, C, dan C++

Proyek Scikit-learn dimulai sebagai scikits.learn, sebuah proyek pada acara Google Summer of Code yang dikembangkan oleh Davin Cournapeau. Namanya berasal dengan ide bahwa *library* tersebut adalah ekstensi dari *library* SciKit (SciPy Toolkit), tapi pengembangannya dilakukan secara terpisah. Kode *library* ini ditulis ulang oleh Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, dan Vincent Michel, yang berasal dari French Institute for Research in Computer Science and Automation di Rocquencourt, Prancis. Tim tersebut membuat rilis public pertama pada 1 Februari 2010 [22].

2.9.3. Matplotlib

Matplotlib merupakan *library* Python yang bertujuan untuk *plotting* data juga sebagai ekstensi dari *library* NumPy. Matplotlib yang ditulis dalam bahasa Python menyediakan API berorientasi objek untuk *plotting* kedalam aplikasi GUI Python. Matplotlib awalnya dikembangkan oleh John D. Hunter pada tahun 2003, tapi selanjutnya pengembangan Matplotlib dilanjutkan oleh komunitas pengembang yang aktif hingga sekarang [23].

2.9.4. Seaborn

Seaborn adalah *library* untuk visualisasi data yang didasarkan pada *library* Matplotlib. Seaborn menyediakan antar-muka tingkat tinggi untuk menggambarkan grafik statistik yang informatif dan menarik. Seaborn juga terintegrasi dengan dekat dengan struktur data pada Pandas. Cara Seaborn bekerja adalah dengan menangkap keseluruhan *dataframe* atau *array* berisikan data dan memetakannya menjadi plot data yang informative [24].

2.10. Jupyter Notebook

Jupyter Notebook adalah aplikasi berbasis web open source yang dapat digunakan untuk membuat dan membagikan dokumen. Dokumen ini berisi kode, persamaan matematika, visualisasi maupun text. Jupyter notebook ini dikelola oleh orang-orang yang tergabung pada Project Jupyter.

Jupyter Notebook merupakan project spin-off dari IPython, yang pada mulanya memiliki proyek tersendiri yaitu Notebook IPython. Memiliki nama Jupyter karena mendukung Bahasa pemrograman Julia, Python dan R. Jupyter disajikan dengan *kernel* IPython, sehingga memungkinkan Anda untuk menulis program dengan menggunakan Python. Namun, pada saat ini ada lebih dari 100 *kernel* lainnya yang dapat digunakan pada Jupyter Notebook [21].

2.11. Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah bahasa pemodelan pada bidang pengembangan dan rekayasa perangkat lunak yang bertujuan untuk memudahkan dokumentasi perangkat lunak dengan cara visualisasi desain dari sebuah sistem.

2.11.1. Use Case dan Use Case Scenario

Usecase Diagram secara grafis menggambarkan interaksi sistem baik sistem internal, sistem eksternal, dan pengguna. Dengan kata lain, untuk mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat.

Terdapat beberapa simbol dalam menggambarkan usecase diagram diantaranya, usecase, aktor, dan relasi. Penamaan usecase didefinisikan secara

sesimpel mungkin, dapat dipahami dengan mudah, dan menggunakan kata kerja. Aktor merupakan segala hal yang berada di luar sistem yang akan menggunakan sistem tersebut tetapi tidak semua aktor itu adalah manusia bisa saja sistem lain yang berinteraksi dengan sistem tersebut [22, 23].

2.11.2. Activity Diagram

Activity diagram adalah diagram yang secara visual merepresentasikan aliran aktivitas yang terjadi pada sebuah sistem. Didalamnya juga dapat berisikan pilihan atau pengulangan. Sebagai bagian dari UML, *activity diagram* dapat memodelkan baik pemrosesan pada komputer maupun proses bisnis. *Activity diagram* merupakan pengembangan dari *Use Case* yang memiliki alur aktivitas.

Diagram aktivitas memiliki komponen dengan bentuk-bentuk tertentu. Komponen tersebut antara lain berbentuk elips untuk menggambarkan aksi, belah ketupat untuk menggambarkan percabangan, dan lingkaran sebagai awal dan akhir *activity diagram*. Setiap aktivitas dihubungkan dengan tanda panah yang terurut mengarah dari awal sampai akhir sesuai urutan aktifitasnya [23].