BAB II

TINJAUAN PUSTAKA

2.1. Konsep Teoritis

2.1.1. *Kecerdasan Buatan (Artificial Intelligence)*

Maksud dari kecerdasan buatan adalah mesin yang memiliki kemampuan seperti manusia, misalnya berpikir, mempertimbangkan kegiatan yang akan dilakukan, dan juga mengambil keputusan saat dihadapkan pada sebuah permasalahan. Berbagai permasalahan yang ditangani oleh kecerdasan buatan yang semakin berkembang memungkinkan kecerdasan buatan untuk membuat cabang-cabang ilmu lainnya (T. Sutojo, S.Si. dkk., 2011). Sehingga oleh karena itu, ilmu dalam kecerdasan buatan secara umum terbagi menjadi 3, yaitu sistem pakar, *fuzzy logic* dan jaringan saraf tiruan.

2.1.1.1. Sistem Pakar

Sistem pakar dirancang layaknya seorang pakar, di mana pakar tersebut memiliki pengetahuan, pengalaman, dan metode khusus untuk menjawab pertanyaan dan memecahkan suatu permasalahan yang diperoleh dari komunikasi dengan pengguna (*user*). Pengetahuan seorang pakar yang digunakan dalam sistem pakar diinput ke dalam komputer, sehingga seseorang yang bukan pakar menggunakan sistem pakar untuk meningkatkan kemampuan memecahkan masalah, sedangkan seorang pakar menggunakan sistem pakar untuk *knowledge assistant*. Fitur lain dari sistem pakar adalah kemampuannya untuk menjelaskan saran atau rekomendasi yang diberikannya. Penjelasan dilakukan dalam subsistem

yang disebut subsistem penjelasan, di mana bagian ini memeriksa penalaran yang dibuatnya sendiri dan menjelaskan operasi-operasinya (T. Sutojo, S.Si. dkk., 2011).

2.1.1.2. *Fuzzy Logic*

Fuzzy logic adalah sebuah metode yang dapat digunakan untuk memecahkan permasalahan, dan sesuai untuk diimplementasikan pada sistem yang sederhana, sistem kecil, *embedded system*, jaringan PC, *multi channel* atau *workstation* berbasis akuisisi data, dan sistem kontrol. Selain dapat diimplementasikan pada sistem, metode ini juga dapat diterapkan pada *hardware*, *software,* maupun perpaduan antara keduanya. Dikatakan bahwa segala sesuatu di dalam logika klasik adalah bersifat biner, jadi dapat diartikan bahwa kemungkinan yang ada hanyalah dua saja, "Ya atau Tidak", "Benar atau Salah", "Baik atau Buruk", dan lain-lain. Macam-macam kemungkinan tersebut menghasilkan nilai keanggotaan 0 atau 1. Berbeda dalam logika *fuzzy*, nilai keanggotaan berada di antara 0 dan 1, sehingga bisa saja sebuah keadaan mempunyai dua nilai "Ya dan Tidak", "Benar dan Salah", "Baik dan Buruk" secara bersamaan, namun besar nilainya tergantung pada bobot keanggotaan yang dimilikinya (T. Sutojo, S.Si. dkk., 2011).

2.1.1.3. Jaringan Saraf Tiruan

Jaringan saraf tiruan (JST) adalah sebuah model untuk mengolah suatu informasi berdasarkan inspirasi sistem saraf secara biologis, dan cara kerjanya menyerupai otak manusia. Dasar dari model ini adalah struktur dari sistem pengolahan informasi yang terdiri dari sejumlah besar elemen pemrosesan yang

saling berhubungan (neuron), bekerja secara bersama-sama untuk menyelesaikan sebuah masalah. Cara kerja JST menyerupai cara kerja manusia, yaitu belajar dari contoh. Contoh dari pengaplikasian JST adalah pengenalan pola atau klasifikasi data melalui proses pembelajaran. Belajar dalam sistem biologis melibatkan penyesuaian terhadap hubungan sinapsis yang ada antara neuron, dan hal tersebut juga berlaku untuk JST (T. Sutojo, S.Si. dkk., 2011).

2.1.2. *Machine Learning*

Machine Learning (ML) atau pembelajaran mesin adalah sebuah teknik inferensi data dengan pendekatan matematis (Putra, 2019). ML juga merupakan pendekatan dalam kecerdasan buatan yang meniru perilaku manusia untuk melakukan pemecahan masalah atau melakukan otomasi. Seperti JST, ML juga memiliki ciri khas yaitu adanya proses pelatihan, sehingga ML membutuhkan data untuk dipelajari (*data training*) (Hania, 2017). ML bergantung pada pola dan kesimpulan, sehingga untuk mendapatkannya, algoritma ML menghasilkan model matematika yang didasari dari sampel data. Data yang digunakan terbagi menjadi 2 kategori, yaitu *data training* dan *data testing*. Data training menjadi bahan pelatihan algoritma dalam mencari model yang sesuai, sedangkan data testing digunakan untuk melakukan pengujian terhadap performa model yang didapatkan pada tahapan evaluasi.

ML melibatkan proses struktural di mana setiap tahap membangun versi mesin yang lebih baik. Untuk penyederhanaan, proses ML terbagi menjadi 3, yaitu data sebagai *input*, abstraksi data, dan generalisasi ("https://inixindojogja.co.id/mengenal-machine-learning", 2017). Data yang

diinput dapat berasal dari *file*, maupun yang berasal dari sebuah *database*. Abstraksi data yang dimaksud adalah sebagai perwakilan dari data dalam bentuk terstruktur berdasarkan algoritma yang dipilih, dan menjadi bagian dari pelatihan dasar. Generalisasi mengacu pada kemampuan model untuk beradaptasi dengan baik terhadap data baru (dengan distribusi yang sama saat membuat model) yang akan diinput kemudian. Langkah-langkah dalam ML adalah :

1. Mengumpulkan Data

   Data yang dikumpulkan (*Dataset*) merupakan data yang bersifat mentah, dapat berupa Ms. Excel, CSV, dan lainnya. Semakin banyak variasi data relevan yang dikumpulan, maka semakin baik prospek pembelajaran mesin tersebut.

2. Mempersiapkan Data

   Kualitas data sangat berpengaruh dalam mengembangkan proses analisis. Sehingga kita harus memperhatikan masalah-masalah pada data, misalnya kehilangan data dan lainnya.

3. Melatih Sebuah Model

   Di dalam langkah ini, pemilihan algoritma dan representasi data yang tepat dalam bentuk model menjadi kedua hal yang ikut terlibat. Data yang disiapkan terbagi menjadi dua, yaitu *training data* yang berperan dalam pengembangan model, dan *data test* sebagai referensi.

4. Mengevaluasi Model

   Tahap ini bertujuan untuk menguji keakuratan, sehingga *data test* digunakan untuk menentukan ketepatan dalam pemilihan algoritma

berdasarkan hasil pengujian. Pengujian pemeriksaan ketepatan model yang lebih baik adalah dengan melihat kinerja pada data yang tidak digunakan sama sekali selama pembuatan model.

5. Meningkatkan Kinerja

   Keterlibatan pemilihan model yang berbeda atau memperkenalkan variabel lainnya di dalam langkah ini bertujuan untuk meningkatkan efisiensi. Oleh karena itu waktu yang banyak sangat dibutuhkan dalam melakukan pengumpulan dan persiapan data.

Metode yang terdapat pada ML secara umum adalah klasifikasi dan prediksi atau regresi. Klasifikasi bertujuan untuk mengklasifikasikan objek berdasarkan ciri tertentu yang membedakan setiap benda. Sedangkan prediksi atau regresi bertujuan untuk memperkirakan output dari input data sesuai dengan data yang telah dipelajari dalam *data training* (Hania, 2017).

2.1.2.1. *Training, Development, Testing Set*

Ada dua istilah penting dalam pembangunan model *machine learning*, yaitu *training* dan *testing*. *Training* merupakan proses pembangunan model, sedangkan *testing* adalah proses pengujian kinerja model pembelajaran. *Dataset* adalah kumpulan data (sampel dalam statistik). Sampel yang dimaksud adalah data yang digunakan untuk membuat model maupun mengevaluasi model ML. Secara umum *dataset* terbagi menjadi 3 jenis yang tidak beririsan (satu sampel pada himpunan tertentu tidak muncul pada himpunan lainnya), yaitu :

1. *Training set*, yang merupakan himpunan data yang berfungsi untuk melatih atau membangun model.
2. *Development set*, yang merupakan himpunan data yang berfungsi untuk mengoptimasi saat melatih model. Model dilatih menggunakan *training set* dan pada umumnya kinerja saat latihan diuji dengan *development set*, sehingga berguna untuk generalisasi (agar model mampu mengenali pola secara generik).
3. *Testing set*, yang merupakan himpunan data yang berfungsi untuk menguji model setelah proses pelatihan berakhir.

2.1.2.2. Algoritma dalam *Machine Learning*

Kinerja ML berubah-ubah sesuai dengan parameter pembelajaran dan diukur oleh fungsi tujuan, yaitu mengoptimalkan nilai fungsi tertentu dengan meminimalkan nilai error, atau meminimalkan nilai loss. Loss adalah ukuran seberapa dekat atau berbeda model yang dihasilkan dengan konsep asli, sedangkan error adalah salah satu cara untuk mengukur loss. Secara intuitif, ML menyerupai seseorang yang sedang belajar. Kesalahan terjadi pada masa awal, namun diperbaiki kemudian dengan diberi tahu mana yang benar lalu menyesuaikan diri secara perlahan agar menjadi benar. Hal tersebut diimplementasikan pada ML dengan mengubah parameter pembelajaran untuk mengoptimalkan suatu fungsi tujuan. Namun ML memerlukan data yang banyak jika dibandingkan dengan seseorang yang dapat belajar dengan contoh yang sedikit.

Dari sisi metode pembelajaran, algoritma ML dapat dikategorikan menjadi *supervised learning*, *semi-supervised learning*, *unsupervised learning*, dan *reinforcement learning*.

1. *Supervised Learning*

   S*upervised learning* adalah pembelajaran terarah atau diawasi. Artinya, pembelajaran ini diumpamakan sebagai guru yang mengajar (mengarahkan) dan siswa yang diajar. Sebagai contoh, seorang guru menuliskan kosakata berbahasa Inggris '*this, that, these, those*' di papan tulis sebagai contoh, kemudian guru tersebut mengajarkan cara melafalkan kata-kata tersebut. Contoh kosakata bahasa Inggris menjadi *input*, dan cara melafalkannya menjadi *desired output*. Pasangan *input-desired output* ini disebut sebagai *instance* (untuk kasus *supervised learning*). Pembelajaran metode ini disebut *supervised* karena ada yang memberikan contoh jawaban (*desired output*). Algoritma yang terdapat pada *supervised learning* adalah *Classification* (*Decision Trees*, *Neural Network* dan *Naives Bayes*, dan lain lain), dan *Regression* (*Linear Regression*, *Logistic Regression*, *Regression Tree*, *Neural Network*, *Forecasting*, dan lain lain).

2. *Semi-supervised Learning*

   Semi-supervised learning mendekati *supervised learning*, namun yang membedakannya adalah pada proses pelabelan data. Jika pada *supervised learning* guru harus membuat kunci jawaban, pada *semi-supervised learning* tidak ada kunci jawaban dari guru namun diperoleh

secara otomatis. Pembelajaran pada *semi-supervised learning* hanya melibatkan sedikit data, kemudian membuat data tambahan menggunakan *supervised* atau *unsupervised learning* dan model belajar dari data tambahan tersebut.

3. *Unsupervised Learning*

   Berlawanan dengan *supervised learning*, pada *unsupervised learning* tidak ada guru yang membimbing. Berbeda dengan *supervised learning* yang memiliki *desired output*, pada *unsupervised learning* tidak ada *desired output* (jelas, tidak ada gurunya, tidak ada yang memberi contoh). Algoritma yang terdapat dalam *unsupervised learning* adalah *Clustering* (*K-means*, *K-nearest-neighbor*).

4. *Reinforcement Learning*

   Mesin dilatih untuk mengambil keputusan spesifik berdasarkan kebutuhan yang bertujuan memaksimalkan kinerja. Inti dari pembelajaran ini adalah mesin melatih dirinya secara terus menerus mengikuti lingkungan yang dipengaruhinya, dan mengimplementasi pengetahuan yang diperoleh untuk menyelesaikan masalah. Juga dipastikan keterlibatan seseorang lebih dikurangi sehingga menghemat lebih banyak waktu. Algoritma yang terdapat dalam *reinforcement learning* adalah *Markov Decision Process* (Nusantara, 2017).

2.1.3. Deret Waktu

2.1.3.1. Deret Waktu dan Data Deret Waktu

Deret waktu adalah serangkaian observasi yang tercatat dalam interval waktu yang teratur. Berdasarkan pada skala observasi yang dilakukan, deret waktu dapat mencakup per jam, harian, mingguan, bulanan, triwulanan, dan tahunan. Terkadang per detik maupun per menit dapat tercakup di dalamnya juga. Analisis terhadap deret waktu dilakukan sebagai persiapan dalam memprediksi deret waktu tersebut. Segala deret waktu dapat terbagi menjadi beberapa komponen, yang terdiri atas tingkat dasar, *trend*, musiman, dan residu (*error*).

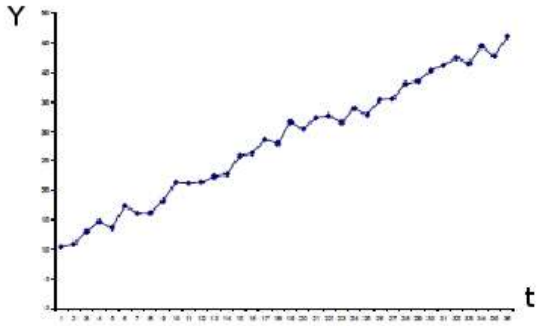
Observasi terhadap trend dilakukan jika ditemukan peningkatan maupun penurunan diagonal pada deret waktu tersebut. Sebagaimana observasi terhadap musiman dilakukan saat terdapat pola berulang berbeda di antara interval reguler yang disebabkan oleh faktor musiman (Prabhakaran, 2019b).

Data deret waktu adalah data yang dikumpulkan dalam kurun waktu yang menggambaran perkembangan suatu observasi yang terjadi dalam kurun waktu tersebut. Berikut ini adalah beberapa variasi gerak atau variasi pad deret waktu, yaitu :

1. Gerak Jangka Panjang (*Trend*)

   Pola gerak ini menggambarkan pergerakan data dalam jangka waktu panjang. Pola gerak ini mencerminkan keadaan yang bersifat kontinuitas, di mana keadaan tersebut memiliki arti bahwa pergerakan tersebut dianggap stabil, karena arahnya yang cenderung naik maupun turun secara terus menerus pada umumnya. *Trend* biasanya digunakan

sebagai prediksi untuk membuat suatu perencanaan. Terdapat 2 jenis *trend*, yaitu *trend* linear (pola garis lurus) dan *trend* non linear (pola lengkung).

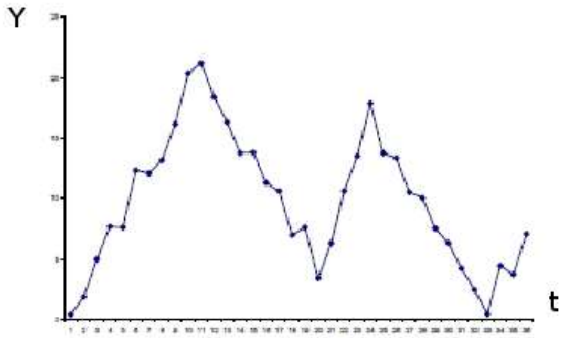


Gambar 2.1 Grafik *Trend*

Sumber : Wanto (2016), Hal. 4

2. Gerak Siklis

Pada pola ini, terdapat variasi jangka panjang di sekitar garis trend dengan tempo yang lebih pendek dan terjadi secara berulang-ulang, namun tidak secara periodik. Terdapat 4 fase kejadian yang terjadi (dalam jangka waktu tertentu) pada pola gerak ini, yaitu kemajuan, kemunduran, depresi dan pemulihan.

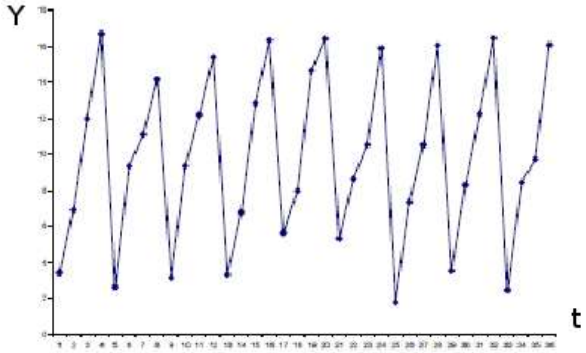


Gambar 2.2 Grafik Siklis

Sumber : Wanto (2016), Hal. 5

3. Gerak Musiman

Pola ini terjadi lebih teratur dan sifatnya lengkap. Misalnya terjadi pada kalender tahunan. Pola ini tetap berlangsung dari waktu ke waktu. Yang menyebabkan terjadinya pola ini adalah iklim dan kebiasaan.

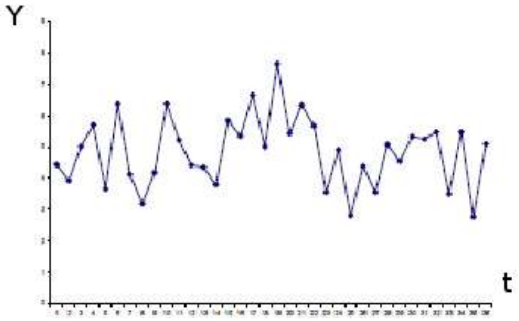


Gambar 2.3 Grafik Musiman

Sumber : Wanto (2016), Hal. 6

4. Gerak Ireguler

Pola gerak ini memiliki sifat yang tidak teratur dan modelnya sulit untuk digambarkan karena faktor perang, bencana alam, dan kekacauan.



Gambar 2.4 Grafik Ireguler

Sumber : Wanto (2016), Hal. 6

2.1.3.2. Stasioneritas dan Nonstasioneritas Data Deret Waktu

Stasioneritas adalah salah satu bagian pada deret waktu. Stasioneritas dalam suatu deret adalah di mana nilai dalam deret bukan merupakan fungsi dalam waktu (Prabhakaran, 2019b). Kestasioneran menjadi salah satu bentuk asumsi yang bertujuan untuk meminimalisir kesalahan dalam permodelan.

Stasioneritas berarti tidak adanya peningkatan maupun penurunan pada data, sehingga fluktuasi data berada pada sekitar nilai rata-rata yang konstan, tidak bergantung dengan waktu dan varians dari fluktuasi tersebut harus konstan setiap waktu. Varians yang tidak stasioner perlu ditransformasi datanya agar menjadi stasioner. Hal serupa juga diaplikasikan pada mean, namun keduanya memiliki metode yang berbeda dalam menstastionerkannya.

2.1.3.3. Pembedaan (Diferensiasi)

Secara umum, metode yang digunakan untuk menstasionerkan mean dari sebuah deret adalah dengan melakukan diferensiasi (minimal sekali). Diferensiasi adalah transformasi yang dilakukan untuk menstasionerkan deret tersebut. Jika Z_t adalah sebuah nilai pada suatu waktu (t), maka diferensiasi pertama adalah $Z = Z_t - Z_{t-1}$. Dapat dikatakan bahwa diferensiasi terhadap deret hanyalah mencari selisih dari observasi yang dikalkulasikan secara matematis untuk membentuk deret waktu yang baru (Rasyidi, 2017). Jika diferensiasi pertama tidak menghasilkan deret yang stasioner, maka dapat dilakukan diferensiasi kedua, dan seterusnya.

Sebagai contoh, terdapat sebuah deret : [1, 4, 2, 10, 16]

Diferensiasi pertama menghasilkan : [4-1, 2-4, 10-2, 16-10] = [3, -2, 8, 6]

Diferensiasi kedua menghasilkan : [-2-3, 8-(-2), 6-8]

2.1.3.4. Fungsi Autokorelasi (ACF)

Pada X_t yang stasioner, $E(X_t) = \mu$ dan $Var(X_t) = \sigma^2$ adalah konstan dan $Cov(X_t, X_s)$ adalah fungsi dari selisih waktu $|t - s|$. Kovarians dan korelasi antara X_t dan X_{t+k} berturut-turut adalah

$$\gamma_k = Cov(X_t, X_{t+k}) = E(X_t - \mu, X_{t+k} - \mu)$$

Rumus 2.1 Fungsi Kovarians

Sumber : Wanto (2016), Hal. 9

dan

$$\rho_k = Corr(X_t, X_{t+k}) = \frac{Cov(X_t, X_{t+k})}{\sqrt{Var(X_t)}\sqrt{Var(X_{t+k})}} = \frac{\gamma_k}{\gamma_0}$$

Rumus 2.2 Fungsi Korelasi

Sumber : Wanto (2016), Hal. 9

Sebagai fungsi dari k, γ_k dinamakan fungsi autokovarians ρ_k dan dinamakan fungsi autokorelasi. Berikut ini yang merupakan sifat dari γ_k dan ρ_k :

1. $\gamma_0 = Var(X_t)$ dan $\rho_0 = 1$
2. $|\gamma_k| \leq \gamma_0$dan $|\rho_k| \leq 1$
3. $\gamma_k = \gamma_{-k}$ dan $\rho_k = \rho_{-k}$

Fungsi autokorelasi dapat dihitung dengan rumus berikut :

$$\hat{\rho}_k = \frac{\sum_{t=1}^{n-k}(X_t - \bar{X})\,(X_{t+k} - \bar{X})}{\sum_{t=1}^{n}(X_t - \bar{X})^2}$$

Rumus 2.3 Fungsi Autokorelasi

Sumber : Wanto (2016), Hal. 10

dengan ρ_k adalah nilai korelasi sampel pada lag-k dan n adalah jumlah observasi.

Signifikansi suatu fungsi autokorelasi dapat diuji dengan membuat plot dari fungsi autokorelasi tersebut dan melakukan perbandingan terhadap error standarnya. Plot tersebut dikenal sebagai *correlogram*, dan untuk membuatnya, perlu dilakukan kalkulasi terhadap nilai error standar dari fungsi autokorelasi tersebut. Error standar ini digunakan untuk melihat apakah fungsi autokorelasi berbeda secara nyata dengan nol. Rumus untuk mengkalkulasikan error standar adalah sebagai berikut :

$$SE_{(\hat{\rho}_k)} = \sqrt{\frac{1 + 2\sum_{i=1}^{i-1} \hat{\rho}_i{}^2}{n}}$$

Rumus 2.4 Fungsi Error Standar

Sumber : Wanto (2016), Hal. 10

dengan $SE_{(\hat{\rho}_k)}$ adalah nilai error standar dari $\hat{\rho}_k$ dan $\hat{\rho}_i{}^2$ adalah nilai autokorelasi sampel pada lag I, dan k adalah selisih waktu.

Pada uji korelasi, H_0 didefinisikan dengan $\rho_k = 0$ yaitu tidak ada autokorelasi, sedangkan H_1 adalah $\rho_0 \neq 0$ yaitu ada autokorelasi antar observasi. Statistik uji yang digunakan dalam uji autokorelasi adalah uji *t* yang dirumuskan sebagai berikut :

$$t_{hit} = \frac{\hat{\rho}_k}{SE_{(\hat{\rho}_k)}}$$

Rumus 2.5 Fungsi Statistik Uji dalam Uji Autokorelasi

Sumber : Wanto (2016), Hal. 10

Dengan $df = n - k$. Daerah penolakan yang digunakan adalah H_0 ditolak jika $|t_{hit}| > t_{\frac{a}{2},df}$. Selain menggunakan uji *t*, plot fungsi autokorelasi dapat digunakan untuk melihat keberadaan autokorelasi antar observasi. Apabila tidak terdapat lag yang keluar dari batas signifikansi, maka dapat disimpulkan bahwa tidak terdapat korelasi antar lag.

2.1.3.5. Fungsi Autokorelasi Parsial (PACF)

Fungsi autokorelasi parsial antara X_t dan X_{t+k} adalah korelasi antara X_t dan X_{t+k} setelah ketergantungan linearnya terhadap X_{t+1}, X_{t+2}, …, X_{t+k-1} dihilangkan. Autokorelasi parsial antara X_t dan X_{t+k}, dinotasikan dengan ϕ_{kk}, dirumuskan sebagai berikut :

$$\phi_{11} = \rho_1$$

$$\phi_{22} = \frac{det(P_2^*)}{det(P_2)} = \frac{\begin{vmatrix} 1 & \rho_1 \\ \rho_1 & \rho_2 \end{vmatrix}}{\begin{vmatrix} 1 & \rho_1 \\ \rho_1 & 1 \end{vmatrix}}$$

$$\phi_{33} = \frac{det(P_3^*)}{det(P_3)} = \frac{\begin{vmatrix} 1 & \rho_1 & \rho_1 \\ \rho_1 & 1 & \rho_2 \\ \rho_2 & \rho_1 & \rho_3 \end{vmatrix}}{\begin{vmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 \\ \rho_2 & \rho_1 & 1 \end{vmatrix}}$$

$$\phi_{kk} = \frac{det(P_k^*)}{det(P_k)}$$

$$= \frac{\begin{vmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{k-2} & \rho_1 \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{k-3} & \rho_2 \\ \rho_2 & \rho_1 & 1 & \cdots & \rho_{k-4} & \rho_3 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ \rho_{k-2} & \rho_{k-3} & \rho_{k-4} & \cdots & 1 & \rho_{k-1} \\ \rho_{k-1} & \rho_{k-2} & \rho_{k-3} & \cdots & \rho_1 & \rho_k \end{vmatrix}}{\begin{vmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{k-1} \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{k-2} \\ \rho_2 & \rho_1 & 1 & \cdots & \rho_{k-3} \\ \vdots & \vdots & \vdots & & \vdots \\ \rho_{k-1} & \rho_{k-2} & \rho_{k-3} & \cdots & 1 \end{vmatrix}}$$

Rumus 2.6 Fungsi Autokorelasi Parsial

Sumber : Wanto (2016), Hal. 12

dengan ρ_k adalah matriks autokorelasi k × k dan $det(P_k) \neq 0$ dan ρ_k adalah autokorelasi pada lag k. Matriks P_k^* adalah matriks P_k dengan kolom terakhir disubstitusi dengan transpose dari ($\rho_1\ \rho_2\ \cdots\ \rho_k$) dan $det(P_k^*) \neq 0$.

2.1.4. *Autoregressive Integrated Moving Average* (ARIMA)

Autoregressive Integrated Moving Average, disingkat dengan ARIMA adalah salah satu algoritma prediksi berdasarkan sebuah ide bahwa informasi mengenai nilai-nilai pada masa lalu yang terdapat dalam deret waktu itu sendiri dapat digunakan untuk memprediksi nilai-nilai pada masa depan. ARIMA juga merupakan sebuah kelas dari model yang menjelaskan sebuah deret waktu berdasarkan nilai-nilai pada masa lalu, yaitu lagnya maupun residual (*error*) prediksi tersebut, sehingga dapat digunakan persamaan untuk memprediksi nilai-nilai pada masa depan.

Model ARIMA dikembangkan oleh George Box dan Gwilym Jenkins pada tahun 1970, sehingga model ini juga dikenal dengan *Box-Jenkins*. Model ini menggambarkan data deret waktu yang bersifat stasioner. Langkah awal yang harus dilakukan sebelum menentukan model ARIMA yang sesuai adalah mengidentifikasi kestasioneran pada mean dan varians data yang digunakan. Jika keduanya tidak stasioner, maka perlu dilakukan pembedaan (diferensiasi) pada mean dan transformasi pada varians terlebih dahulu.

Menurut Prabhakaran (2019), sebuah model ARIMA dikategorikan menjadi 3 orde, yaitu :

1. '*p*' adalah orde pada AR (*Autoregressive*), yang mengarah pada jumlah lag pada *Z* yang digunakan sebagai prediktor.
2. '*q*' adalah orde pada MA (*Moving Average*), yang mengarah pada jumlah error keterlambatan prediksi yang harus diarahkan ke model ARIMA.
3. '*d*' adalah jumlah minimal diferensiasi yang harus dilakukan untuk membuat deret menjadi stasioner. Jika deret waktu tersebut sudah stasioner, maka *d* = 0.

2.1.4.1. Klasifikasi Model ARIMA

2.1.4.1.1. Model *Autoregressive* (AR) (*p*, 0, 0)

Menurut Prabhakaran (2019), sebuah model AR yang murni adalah di mana X_t hanya bergantung pada lagnya sendiri, sehingga X_t adalah sebuah fungsi dari 'lag dari X_t'. Model ini berawal dari George Udny Yule yang memperkenalkannya pada tahun 1926, kemudian dikembangkan oleh Gilbert Walker pada tahun 1931.

$$X_t = \mu + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \epsilon_t$$

Rumus 2.7 Fungsi AR
Sumber : Wanto (2016), Hal. 14

di mana, X_{t-1} adalah lag pertama pada deret, ϕ_1 adalah parameter AR pertama yang diestimasi oleh model dan μ adalah konstanta yang diestimasi oleh model tersebut, dan ϵ_t adalah *error* pada saat ke-*t*.

2.1.4.1.2. Model *Moving Average* (MA) (0, 0, *q*)

Menurut Prabhakaran (2019), sebuah model MA yang murni adalah di mana X_t hanya bergantung pada keterlambatan *error* prediksi. Proses MA menyatakan

hubungan ketergantungan antara nilai observasi dengan nilai-nilai *error* secara berurutan dari periode *t* hingga $t - q$.

$$X_t = \mu + \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - \cdots - \theta_q\epsilon_{t-q}$$

Rumus 2.8 Fungsi MA

Sumber : Wanto (2016), Hal. 14

di mana θ_1, θ_2, …, θ_q adalah koefisien parameter dari MA.

2.1.4.1.3. Model *Autoregressive Moving Average* (ARMA)

Model ARMA adalah sebuah campuran antara model AR maupun MA (Risqia, 2017). Orde *p* dan *q* dalam model ARMA dapat ditulis menjadi (*p*, 0, *q*), dan persamaannya adalah sebagai berikut:

$$X_t = \mu + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - \cdots - \theta_q\epsilon_{t-q}$$

Rumus 2.9 Fungsi ARMA

Sumber : Wanto (2016), Hal. 15

dengan ϕ_p adalah parameter AR dan θ_q adalah parameter MA.

2.1.4.1.4. Model *Autoregressive Integrated Moving Average* (ARIMA)

Sebuah model ARIMA adalah di mana deret waktu telah didiferensiasi minimal sekali sehingga menjadi stasioner. Ditegaskan kembali, jika data deret waktu tidak stasioner, perlu dilakukan diferensiasi. Dalam membangun model ARIMA, masing-masing nilai *p*, *q*, dan *d* harus diidentifikasi terlebih dahulu (Prabhakaran, 2019).

$$X_t = \mu + (1 + \phi_1)X_{t-1} + (\phi_2 - \phi_1)X_{t-2} + \cdots + (\phi_p - \phi_{p-1})X_{t-p} - \phi_p X_{t-p-1} + \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - \cdots - \theta_q\epsilon_{t-q}$$

Rumus 2.10 Fungsi ARIMA

Sumber : Sumarjaya (2015), Hal. 45

2.1.4.1.5. Model *Integrated Moving Average* (IMA)

Model IMA adalah sebuah model yang sering ditemui dalam permasalahan di bidang ekonomi maupun bisnis. Model ini sendiri memiliki orde (0, *d*, *q*), dan contoh orde yang sering ditemui dalam permasalahan di bidang tersebut adalah (0, 1, *q*).

$$X_t = \mu + X_{t-1} + \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - \cdots - \theta_q\epsilon_{t-q}$$

Rumus 2.11 Fungsi IMA (0, 1, *q*)

Sumber : Sumarjaya (2015), Hal. 46

2.1.4.1.6. Model *Autoregressive Integrated* (ARI)

Model ini memiliki orde (*p*, *d*, 0).

$$X_t = \mu + (1 + \phi_1)X_{t-1} + (\phi_2 - \phi_1)X_{t-2} + \cdots + (\phi_p - \phi_{p-1})X_{t-p} - \phi_p X_{t-p-1} + \epsilon_t$$

Rumus 2.12 Fungsi ARI (*p*, 1, 0)

Sumber : Sumarjaya (2015), Hal. 46

2.1.4.2. Identifikasi Model ARIMA

Untuk membentuk suatu model ARIMA, plot dari data deret waktu perlu dibuat terlebih dahulu, sehingga dapat diketahui apakah pola data deret waktu tersebut mengikuti pola horizontal, trend, siklis, ataupun pola musiman (seasonal). Plot deret waktu ini digunakan untuk menyelidiki stasioneritas data deret waktu

(Wanto, 2016). Jika data deret waktu tersebut belum stasioner, maka perlu dilakukan diferensiasi. Diferensiasi dilakukan pada lag 1, lag 2, dan seterusnya hingga stasioner. Model ARIMA dinyatakan dengan orde (p, d, q), dan berikut ini adalah tabel identifikasi model ARIMA (Syahrir, 2017) :

Tabel 2.1 Perbandingan ACF dan PACF pada Tiga Model

	Sampel ACF	Sampel PACF
AR (p)	• Menurun secara eksponensial dan sinusoidal • Selang-seling (kanan dan kiri)	• Terpotong setelah lag ke-p
MA (q)	• Terpotong setelah lag ke-q	• Menurun secara eksponensial dan sinusoidal • Selang-seling (kanan dan kiri)
ARMA (p, q)	• Terpotong setelah lag ke-q • Menurun secara eksponensial dan sinusoidal dari lag ke-q • Selang-seling (kanan dan kiri) dari lag ke-q	• Terpotong setelah lag ke-p • Menurun secara eksponensial dan sinusoidal dari lag ke-p • Selang-seling (kanan dan kiri) dari lag ke-p

2.1.4.2.1. Identifikasi Orde Pada Diferensiasi (*d*)

Diferensiasi pada deret perlu dilakukan dengan hati-hati, karena jika dilakukan secara berlebih dapat mempengaruhi parameter pada model yang dihasilkan. Orde pada diferensiasi yang benar adalah jumlah diferensiasi minimum yang diperlukan agar plot dari fungsi autokorelasi mencapai angka nol dalam waktu yang singkat.

Jika autokorelasi yang dihasilkan adalah positif untuk jumlah lag yang banyak (≥ 10), maka deret tersebut perlu dilakukan diferensiasi lebih lanjut. Di sisi lain, jika lag pertama dari autokorelasi tersebut bernilai negatif sangat besar, maka deret tersebut mungkin saja terdiferensiasi secara berlebih.

Jika pada proses ini tidak dapat ditentukan dari dua orde diferensiasi, maka pilihlah orde yang menghasilkan standar deviasi yang lebih rendah di dalam deret yang terdiferensiasi (Prabhakaran, 2019).

2.1.4.2.2. Identifikasi Orde Pada AR (*p*)

Jumlah orde pada AR dapat diketahui dengan menginspeksi plot autokorelasi parsial (PACF). Autokorelasi dalam deret yang stasioner dapat diperbaiki dengan menambahkan orde yang cukup pada AR (*p*), sehingga orde pada AR menjadi setara dengan banyaknya lag yang melewati batas signifikansi dalam plot PACF (Prabhakaran, 2019).

2.1.4.2.3. Identifikasi Orde Pada MA (*q*)

Seperti yang dilakukan untuk mengidentifikasi orde pada AR, orde pada MA dapat dicari melalui plot, tetapi melalui plot autokorelasi (ACF). Sebuah MA itu sendiri secara teknis adalah error dari keterlambatan prediksi tersebut. Fungsi autokorelasi menjelaskan jumlah orde pada MA yang diperlukan untuk menghilangkan autokorelasi dalam deret yang stasioner (Prabhakaran, 2019).

2.1.5. Bahasa Pemrograman Python

2.1.5.1. Tentang Python

Python adalah bahasa pemrograman yang diciptakan oleh Guido Van Rossum yang saat itu bekerja sebagai peneliti di Universitas Amsterdam, Belanda. Python kemudian dirilis pertama kalinya pada tahun 1990 dan lisensinya yang *open source* dipegang oleh *Python Software Foundation* (PSF), sebuah lembaga yang didirikan oleh Guido bersama dengan rekan-rekannya di awal tahun 2000.

Berikut ini adalah keunggulan Python dibandingkan dengan bahasa pemrograman lainnya (Adriyan, 2019) :

1. Bersifat *open source*, di mana lisensi python tidak harus dibayar jika Python digunakan untuk membuat aplikasinya, maupun untuk tujuan komersial.
2. Kemudahaan dalam penelitian dan pembacaannya, di mana sintaksnya hampir mendekati bahasa manusia (bahasa Inggris), sehingga penelitian kode program menjadi lebih mudah. Juga pembaca program yang bukan

pembuat program tersebut dapat mengerti kode-kode yang tertulis di dalamnya.

3. Berjenis interpreted, berbeda dengan C , C++, maupun Fortran yang berjenis compiled.
4. Selain pemrograman fungsional, juga mendukung pemrograman berorientasi objek.
5. Sifat kemudahan di dalam penelitian dan pembacannya menyebabkan Python dapat digunakan sebagai bahasa perintah (*command*) untuk mengakomodir aplikasi yang dibuat dengan bahasa pemrograman lainnya.
6. Package yang cukup banyak dapat digunakan untuk komputasi saintifik (*scientific computing*) oleh *scientist* dan *engineer*, *artificial intelligence*, *machine learning*, dan *data science*.
7. Python diimplementasikan pada *search engine* Google, Dropbox, Youtube, NASA, Lucas Film, dan perusahaan besar lainnya.

2.1.5.2. Versi Python

Dua versi utama pada Python adalah Python 2 dan Python 3. Umumnya Python dirilis dengan versi kode *major*, *minor*, dan *micro*. *Macro* yang berarti menunjukkan perubahan besar dalam bahasanya. *Minor* menunjukkan perubahan kecil, misalnya dalam perbaikan penelitian dan kemampuan eksekusi. Dan *micro* lebih tertuju pada perbaikan *bug*. Secara keseluruhan, *user* (*end-user*) Python menggunakan Python yang dirilis stabil berdasarkan penamaan ketiga versi tersebut (Adriyan, 2019).

2.1.5.3. *Python Distribution*

Jumlah *package* pada Python yang sangat banyak menyebabkan beberapa perusahaan teknologi dunia mem*bundle* Python dengan *package* tertentu, dengan tujuan agar user tidak repot dalam melakukan instalasi tambahan setelah proses instalasi Python. Berikut ini adalah beberapa *package* yang didistribusikan kepada *user* (Adriyan, 2019):

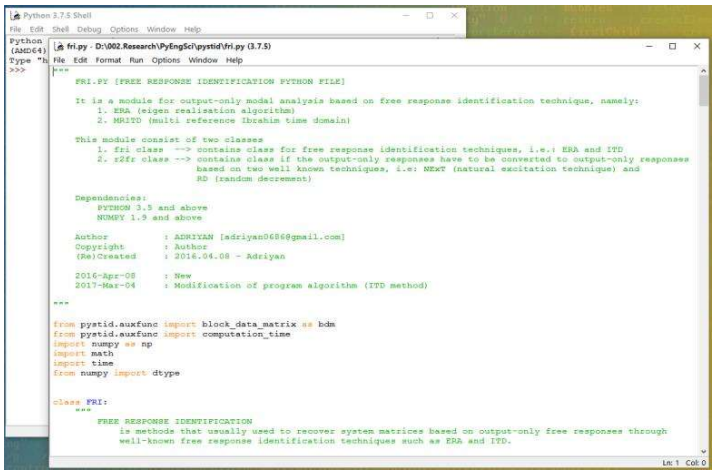
1. Scientific Python Stacks (scipy *stacks*), yaitu kumpulan *package* yang berisi pustaka untuk keperluan *scientific* seperti NumPy (numerical python) untuk tipe data *array* seperti Matlab, Scipy (*scientific python*) untuk perhitungan saintific, Matplotlib untuk *plotting*, SymPy (*Symbolic Python*) untuk komputasi simbolik seperti Maple atau Mathematica, Pandas untuk manipulasi data dan analisis data. Akhirnya adalah IPython sebagai command editor Python secara interaktif dan juga bangunan dasar untuk Jupyter Notebook.
2. *Database*, seperti SQL Alchemy disamping bawaan Python sendiri yaitu SQLite.
3. IDE seperti Spyder selayaknya replika tampilan Matlab (aplikasi komersial).
4. *Notebook* seperti Jupyter Notebook atau Jupyter Lab, yaitu integrasi kode, tulisan (*text*), dan visualisasi pada satu halaman dengan menggunakan *browser* seperti Chrome, Firefox, atau Safari. *Package* ini seperti selayaknya menggunakan Mathematica (aplikasi komersial)

5. *Plotting* atau visualisasi, seperti vtk, MayaVi, Bokeh, HoloView, Seaborn, dan Matplotlib.
6. Aplikasi saintific untuk *domain* tertentu seperti bioinformatics (biopython), *machine learning* (Scikit-learn), image processing (scikit-image), astrofisika (astropy).
7. Untuk membuat antarmuka tampilan secara grafis (*graphical user interfaces* yang disingkat GUI), *user* dapat menggunakan pyqt (Qt-*based*). Bawaan Python untuk GUI ini adalah tk.
8. *Package management*, seperti conda dan *enthought deployment manager* (EDM) disamping bawaan Python sendiri yaitu pip.

2.1.5.4. IDE (*integrated development environment*)

Integrated development environment (IDE) adalah sebuah lingkungan (*environment*) yang digunakan untuk mengembangkan program komputer terintegrasi, baik secara personal maupun berkelompok karena fasilitas-fasilitas yang cukup banyak tersedia, sehingga produktifitas terhadap pengelolaan pengembangan perangkat lunak meningkat. Berikut ini beberapa IDE yang bisa digunakan dalam Python *programming* (Adriyan, 2019):

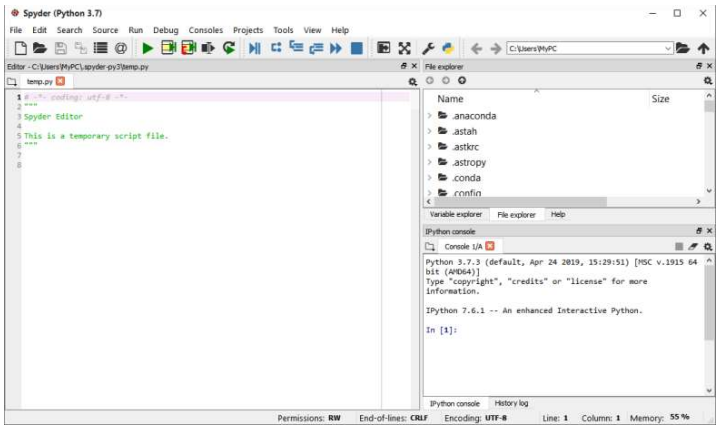
1. IDLE (*integrated development environment*) sebagai IDE bawaan Python setelah proses instalasi selesai di setiap sistem operasi.



Gambar 2.5 Tampilan pada IDLE

Sumber : Adriyan (2019), Hal. 10

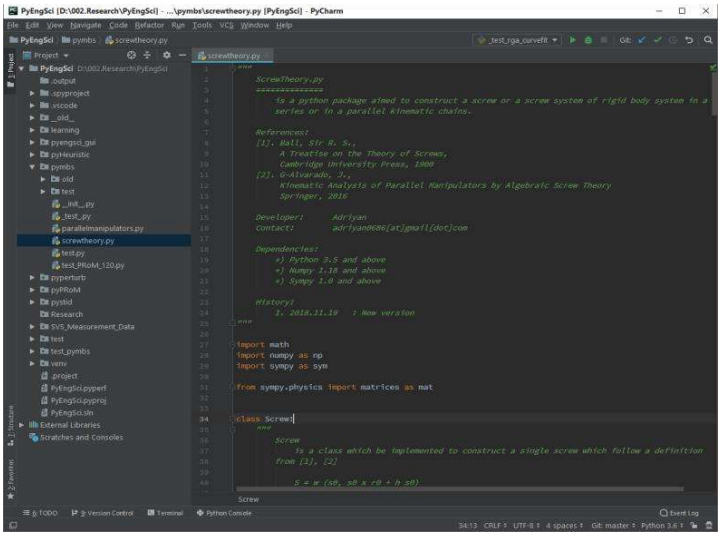
2. Spyder (*Scientific Python Development Environment*) yang dibuat dengan bahasa Python dapat diinstall melalui pip, suatu *pacakage management* bawaan Python. Spyder memiliki lisensi MIT (salah satu lisensi *open source*) dengan GUI seperti Matlab.



Gambar 2.6 Tampilan pada Spyder

Sumber : Adriyan (2019), Hal. 10

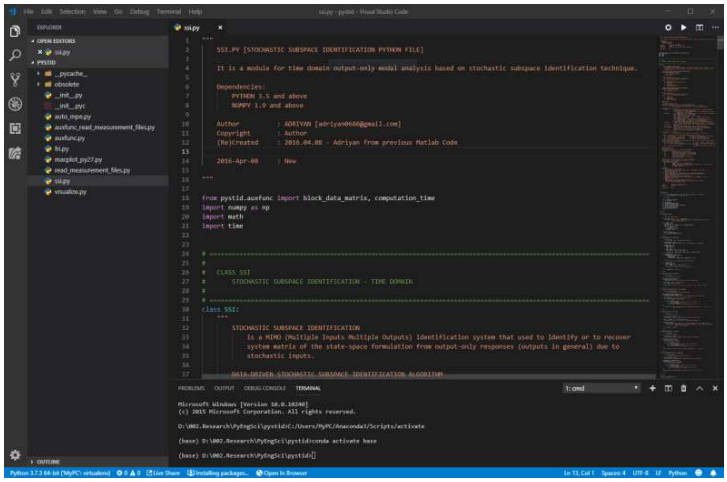
3. PyCharm yang dirilis oleh JetBrains Inc. yang menyediakan versi *community* (*PyCharm Community*) dan profesional (*PyCharm Professional*). *PyCharm Community* adalah gratis bagi pengguna umum, sementara *PyCharm Professional* adalah berbayar. Untuk akademisi atau peneliti, *PyCharm Profesional* gratis selama 1 tahun dengan cara registrasi melalui email institusi dan dapat diperpanjang. Keduanya memiliki tampilan antarmuka yang sama dengan perbedaan hanya terletak pada fitur yang disediakan.



Gambar 2.7 Tampilan pada PyCharm

Sumber : Adriyan (2019), Hal. 11

4. Visual Studio Code (disingkat VS Code) suatu IDE *open source* dari Microsoft yang mendukung berbagai jenis bahasa pemograman melalui instalasi *extensions* tertentu untuk setiap bahasa pemograman. Agar dapat digunakan untuk pemograman python ada beberapa *extensions* VS Code untuk Python yang harus diinstalasi terlebih dahulu.



Gambar 2.8 Tampilan pada VS Code

Sumber : Adriyan (2019), Hal. 12

2.1.5.5. *Package Management*

Package Management adalah sebuah library Python yang bertujuan untuk mengelola *package* yang akan diinstall ke dalam Python *environment* dengan keterikatannya terhadap package lainnya maupun Python tersebut, yang di mana keterikatannya dikaitkan dengan kesesuaian versi yang dikembangkan oleh *developer*-nya dengan Python dan package tertentu. Berikut ini merupakan *package management* yang terdapat pada Python (Adriyan, 2019):

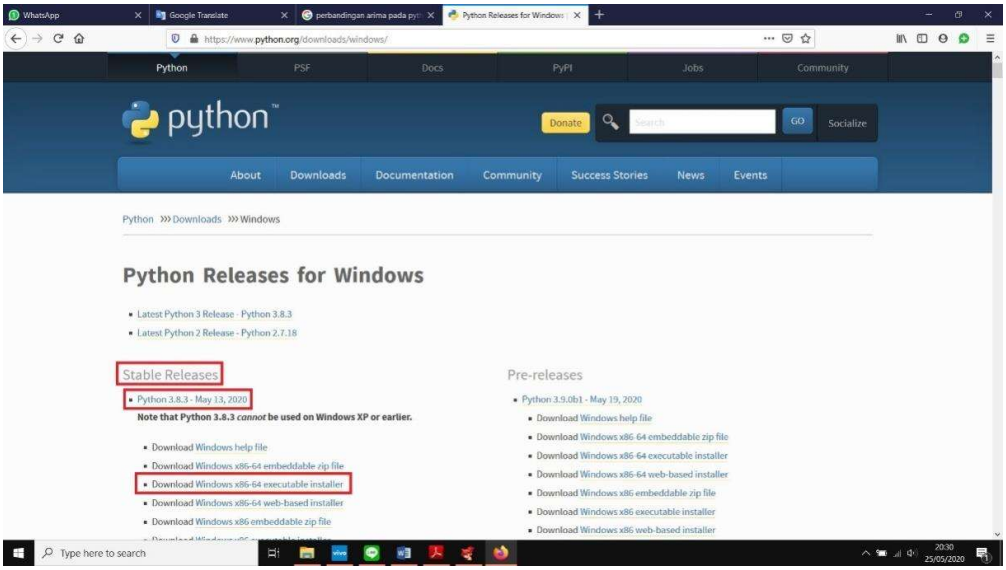
1. pip (*Pip Install Packages*) yang menjadi bawaan dari instalasi Python. Instruksi dalam cmd untuk menginstal sebuah *package* adalah 'pip install nama_package'.
2. *conda package manager* yang dikembangkan oleh Anaconda Inc., bersifat lebih unggul kemampuannya dibandingkan dengan pip, di mana memiliki beberapa fungsi yaitu untuk menginstal package, men-*delete*

package, *create*/*delete environment*, mencari *package* yang sesuai, membersihkan *package* dan *cache* dari lingkungan Python, melakukan *update package*, dan sebagainya.

2.1.5.6. Instalasi Python

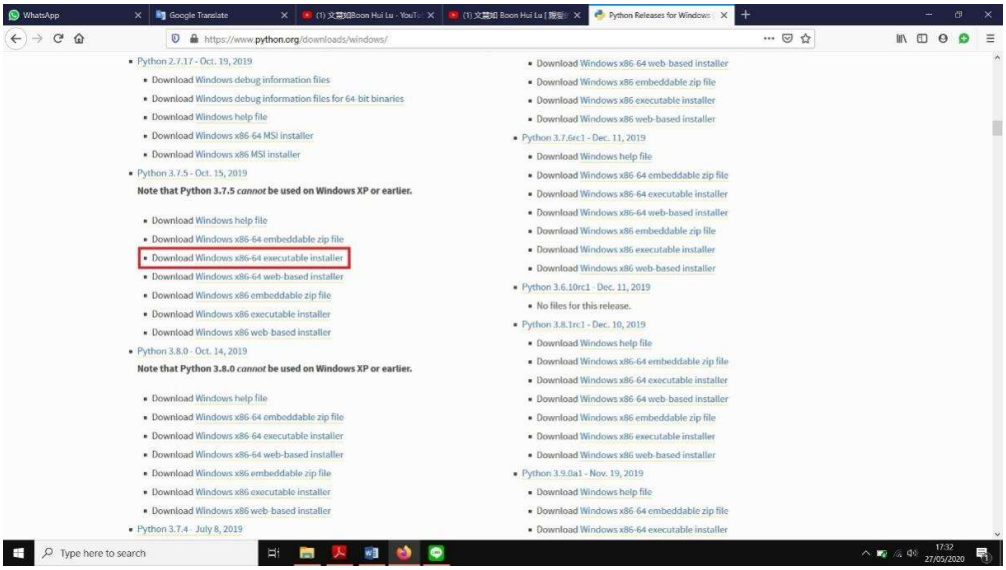
Karena versi Python yang digunakan oleh peneliti adalah Python 3, maka akan dijelaskan instalasi terhadap Python 3. Sistem operasi yang digunakan oleh peneliti adalah Windows 10, namun cara instalasi ini dapat juga diikuti oleh Windows lainnya (Windows 8.1/8/7). Terlebih dahulu terdapat beberapa periapan yang harus dilakukan sebelum menginstal Python :

1. *Installer* Python 3 dapat di*download* melalui *link* berikut : https://www.python.org/downloads/windows/.
2. Kunjungi link tersebut, pada sebelah kiri terdapat *stable release*, yang merupakan versi terbaru dari Python. Perhatikan pada versi terbarunya dan tanggalnya, terdapat berbagai macam *installer* (juga terdapat pada tiap versi) yang dapat di*download*, pilihlah yang *executable-installer* dan secara otomatis sistem akan mendownload file *installer* tersebut. Akan tetapi Python yang digunakan oleh peneliti adalah versi 3.7.5 sesuai dengan panduan dari (Adriyan, 2019), sehingga pada gambar *screenshot* akan ditunjukkan yang versi 3.7.5.



Gambar 2.9 *Stable Releases* pada Python

Sumber : Data Penelitian (2020)



Gambar 2.10 Installer untuk Python 3.7.5

Sumber : Data Penelitian (2020)

Berikut ini adalah langkah-langkah dalam proses instalasi Python 3 :

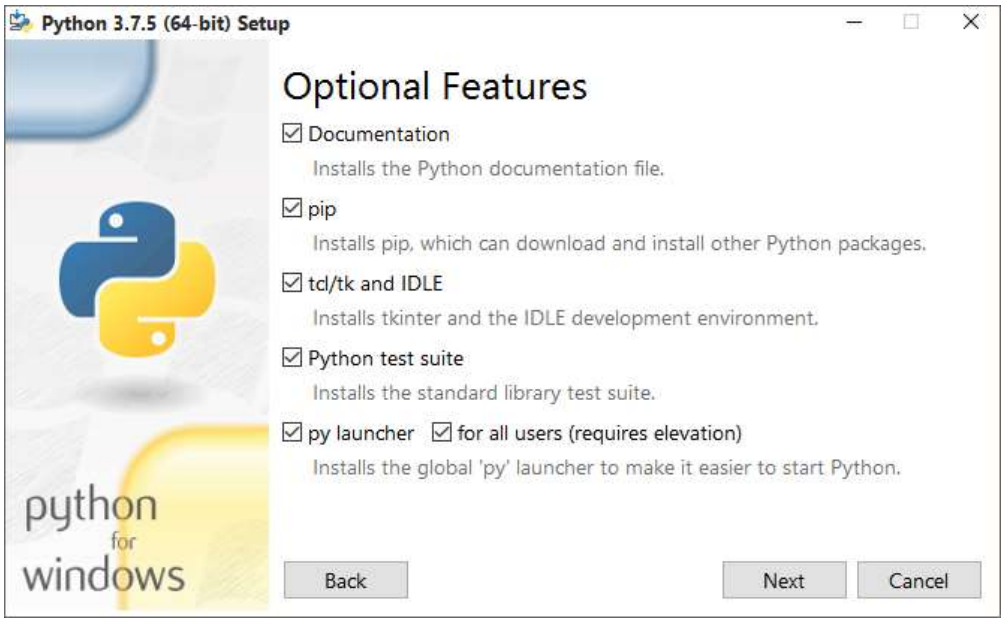
1. Carilah di mana file installer Python tersimpan, eksekusilah file installer tersebut, sehingga muncullah kotak dialog instalasi Python seperti di bawah ini. Beri tanda *checklist* pada '*Add Python 3.7.5 to PATH*' agar direktori Python tersebut ditambahkan sehingga dikenali langsung nantinya oleh sistem operasi Windows.

Gambar 2.11 Tampilan Awal saat Instalasi Python

Sumber : Adriyan (2019), Hal. 17

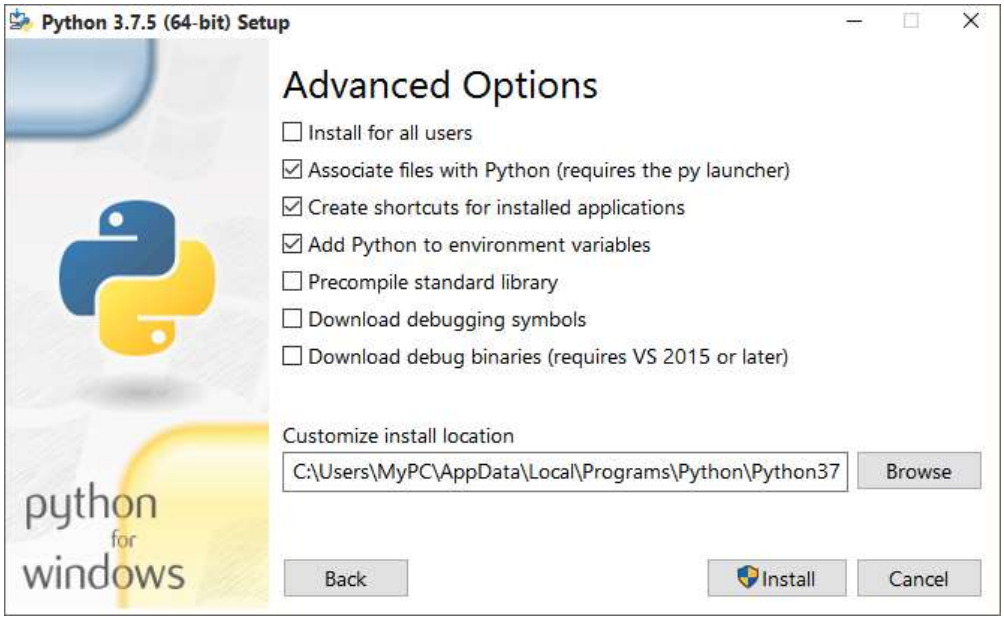
Dengan demikian, ketika 'python' diketikkan melalui *Command Prompt* atau *Windows Power Shell* akan langsung dikenali oleh sistem operasi. Secara *default*, Python itu sendiri juga sudah dilengkapi dengan IDLE, pip dan dokumentasi bahasa Python itu sendiri. Jika '*Add Python 3.7.5 to PATH*' tidak di*checklist*, maka perlu dilakukan konfigurasi tambahan setalah proses instalasi selesai dilakukan.

2. Selanjutnya terdapat 2 pilihan di atas yaitu '*Install Now*' dan '*Customize installation*'. Jika memilih '*Customize installation*', maka dapat dipilih fitur-fiturnya. Tetapi akan lebih baik jika pilihan-pilihan terhadap fitur-fitur dibiarkan saja apa adanya. Jika memilih '*Install Now*', maka fitur-fitur yang di*install* adalah yang sudah di*checklist* secara *default* (seperti saat memilih '*Customize installation*'). Klik tombol *Install*, pada kotak dialog yang ke dua di bawah ini.



Gambar 2.12 Pilihan Fitur pada saat Instalasi Python

Sumber : Adriyan (2019), Hal. 18



Gambar 2.13 Pilihan Lanjutan saat Instalasi Python

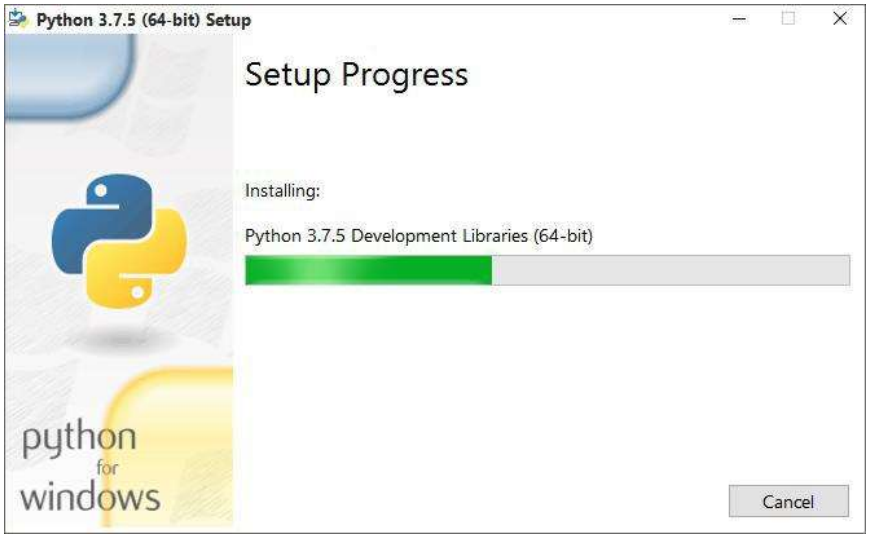
Sumber : Adriyan (2019), Hal. 18

Kemudian akan muncul sebuah kotak dialog yang meminta persetujuan untuk meng*install* Python ke System C pada PC, klik tombol *Yes* untuk melanjutkan ke proses instalasi.

Gambar 2.14 Izin untuk Instalasi Python pada PC

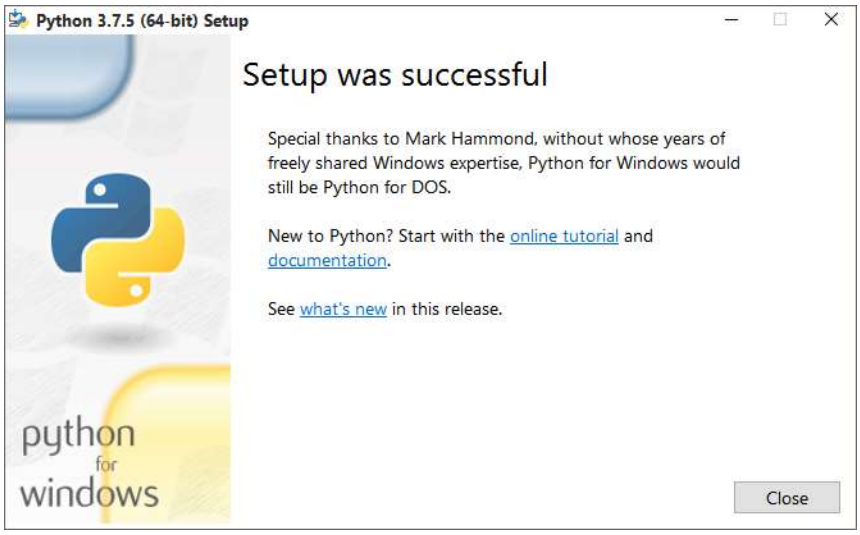
Sumber : Adriyan (2019), Hal. 19

3. Setelah ini akan muncul kotak dialog proses instalasi Python '*Setup Progress*'. Jika proses instalasi sudah selesai, maka akan muncul kotak dialog '*Setup was succesful*'. Klik tombol Close untuk mengakhiri proses instalasi yang juga menutup kotak dialog tersebut.



Gambar 2.15 Proses Instalasi Python

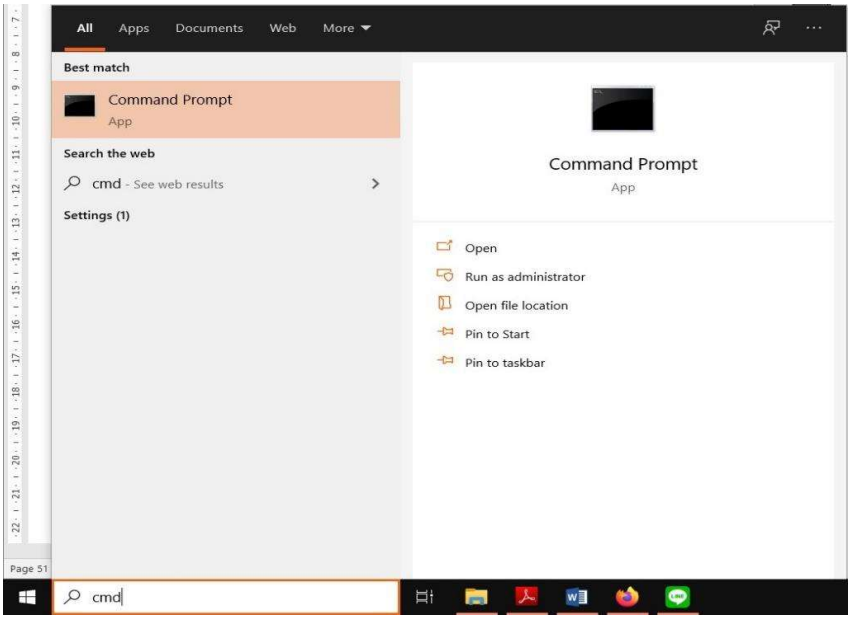
Sumber : Adriyan (2019), Hal. 19



Gambar 2.16 Instalasi Python Berhasil

Sumber : Adriyan (2019), Hal. 20

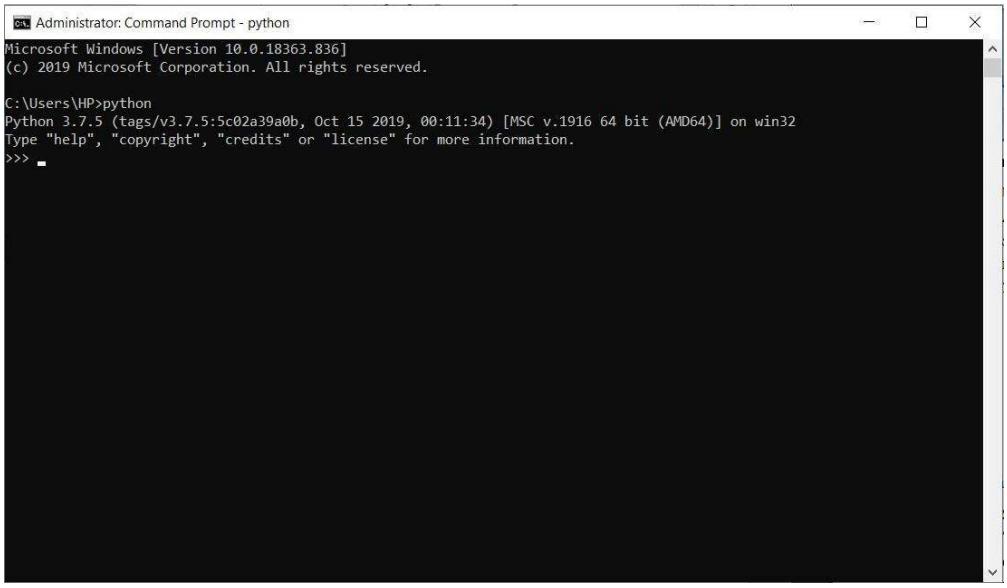
4. Untuk mengecek Python sudah ter*install* dan dikenali oleh sistem operasi pada PC, bukalah *Command Prompt* windows dengan klik *Start Menu*, kemudian ketikkan ‘cmd’ (tanpa tanda petik) dan tekan enter pada *keyboard*.



Gambar 2.17 Membuka CMD Melalui Start Menu

Sumber : Data Penelitian (2020)

5. Setelah jendela *Command Prompt* terbuka, ketikkan ‘python’ (tanpa tanda petik) setelah tanda ”C:\Users\xxx>”, kemudian tekan Enter sehingga akan tampil seperti yang ditunjukkan oleh gambar di bawah ini. “xxx” merupakan nama user account di masing-masing PC. Gambar di bawah ini menjelaskan bahwa Python 3.7.5 64-bit telah ter*install* dan dikenali oleh sistem operasi PC, Windows 10. Ketikkan ‘exit()’ (tanpa tanda petik) setelah tanda “>>>” untuk keluar dari command python, atau klik tutup jendela ini melalui tanda × di kanan atas jendela.

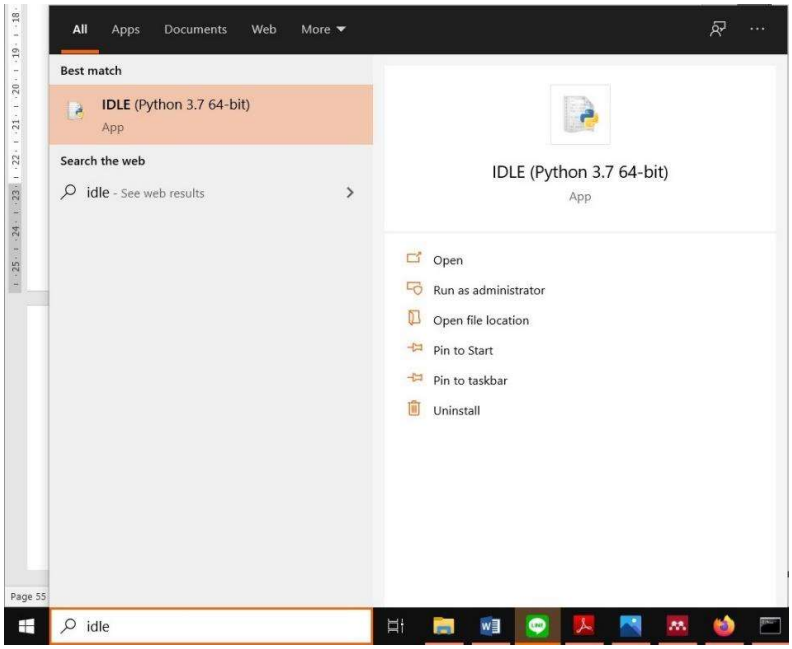


Gambar 2.18 Mengecek Versi Python pada CMD

Sumber : Data Penelitian (2020)

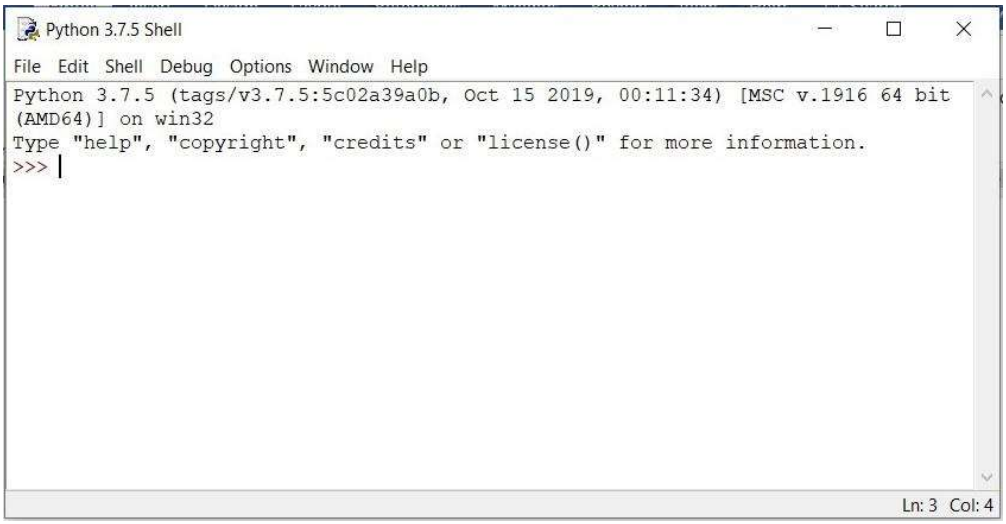
Untuk memulai penelitian kode program pada Python, dapat dilakukan dengan cara :

1. Klik Start Menu, dan untuk mengeksekusi editornya dapat langsung mengetikkan ‘idle’ (tanpa tanda petik). Kemudian terdapat dua pilihan, yaitu bisa mengklik langsung icon IDLE yang baru saja muncul pada hasil search, ataupun menekan tombol enter pada *keyboard*, sehingga muncullah layar aplikasi *Python 3.7.5 Shell*.



Gambar 2.19 Membuka IDLE melalui Start Menu

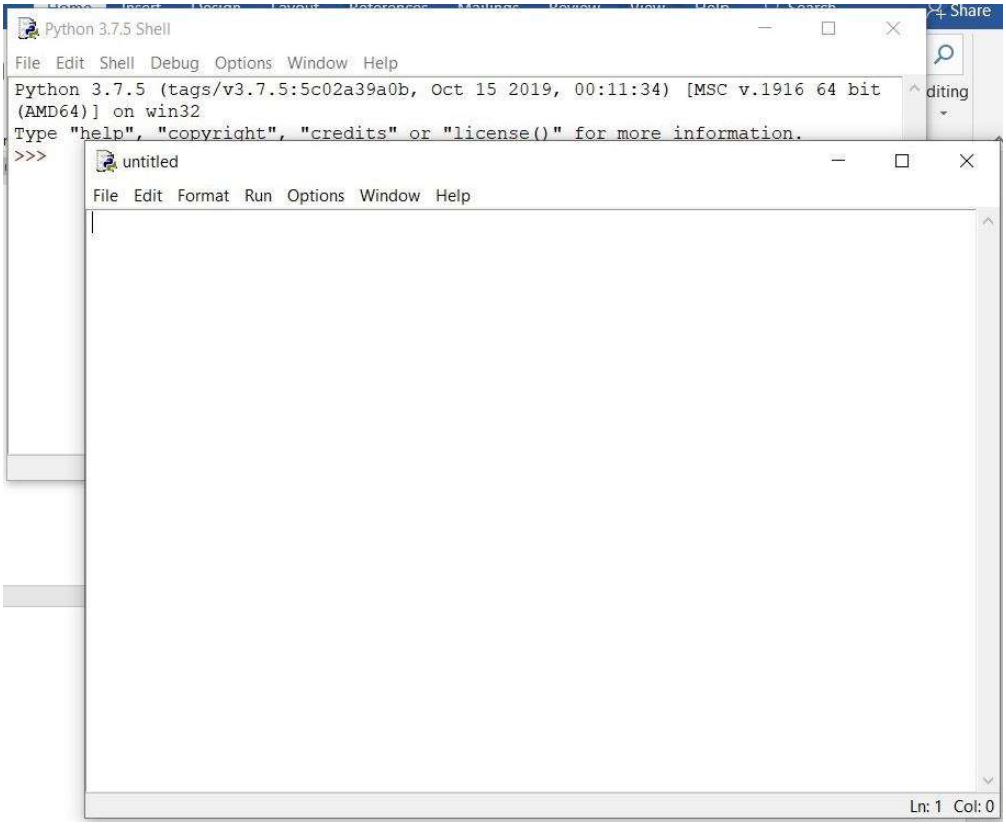
Sumber : Data Penelitian (2020)



Gambar 2.20 Tampilan IDLE

Sumber : Data Penelitian (2020)

2. Klik menu *File* di *Python 3.7.5 Shell* dan pilih *New File* (Ctrl + N) untuk membuka IDLE bawaan Python 3.7.5.



Gambar 2.21 Tempat untuk Mengetikkan Kode Program

Sumber : Data Penelitian (2020)

3. Kode program pada Python dapat langsung diketikkan pada layar tersebut, dan untuk mengeksekusinya klik menu *Run*, lalu klik *Run Module*. Sebelum kode program tersebut dieksekusi, terlebih dahulu sistem akan meminta untuk menyimpan (*save*) *file* kode program tersebut.

2.1.6. Persediaan pada Storage (*Supply in the Storage*)

Persediaan adalah segala sesuatu termasuk sumber daya dari sebuah perusahaan yang disimpan dengan tujuan antisipasi dalam memenuhi permintaan. Terdapat juga persediaan optimal, yang memiliki arti bahwa jumlah barang yang disimpan adalah tingkat yang terbaik dan menguntungkan (Lahu & Sumarauw, 2017). Selanjutnya pengendalian persediaan (supply control) adalah suatu kegiatan yang ditujukan agat persediaan yang ada tidak menipis dan dijaga tingkat optimalnya sehingga biaya persediaan tetap stabil (Indah dkk., 2018).

2.2. Penelitian Terdahulu

Berikut ini adalah beberapa penelitian terdahulu yang dapat digunakan sebagai acuan dalam penelitian yang digagas oleh peneliti :

1. Prediksi Harga Bahan Pokok Nasional Jangka Pendek Menggunakan ARIMA

   Penelitian yang dilakukan oleh Rasyidi (2017) adalah prediksi dengan waktu jangka pendek, yaitu selama sebulan atau tiga puluh hari. Data yang digunakan untuk pemrosesan dalam model ARIMA adalah data yang didapatkan dari Kementrian Perdagangan Republik Indonesia, yang terdiri atas harga bahan pokok dari bulan Mei 2012 hingga bulan Juli 2017, dan terdapat dua belas bahan pokok yang akan diprediksi harganya. Di dalam data harian tersebut, masih terdapat beberapa data harian yang hilang, sehingga dilakukan *data processing* dengan estimasi metode interpolasi linear. Model ARIMA yang digunakan untuk memprediksi harga bahan pokok diimplementasikan dalam bahasa pemrograman R,

dengan menggunakan fungsi *auto.arima* (*package* khusus prediksi), di mana model ARIMA tersebut ditentukan berdasarkan orde *p*, *d* dan *q* yang terbaik. Hasil yang didapatkan adalah error dengan rata-rata sebesar 2.22%, yaitu dengan error rata-rata terkecil 0.34% (kedelai impor), dan error rata-rata terbesar 6.93% (cabai merah biasa). Di antara hasil prediksi tersebut, terdapat beberapa bahan pokok yang tingkat akurasinya rendah pada suatu masa (pada hari raya penanggalan Hijriah) karena dipengaruhi oleh momen puasa, Idul Fitri, dan juga Idul Adha, di mana permintaan terhadap bahan pokok dengan persediaan yang ada adalah tidak seimbang, sehingga terjadi kenaikan terhadap harga bahan pokok dilakukan untuk mengendalikan pasar.

2. Prediksi Kerawanan Wilayah Terhadap Tindak Pencurian Sepeda Motor Menggunakan Metode (S)ARIMA Dan CART

   Sesuai dengan judul penelitiannya, penelitian yang dilakukan oleh (Utomo dan Azhari (2017) mengkombinasikan metode ARIMA dan CART pada wilayah Provinsi DI Yogyakarta dan Kota Yogyakarta itu sendiri. ARIMA digunakan untuk memprediksi nilai periode pada bulan Januari 2016 hingga bulan Juni 2016, sedangkan CART digunakan untuk mengklasifikasikan kerawanan wilayah terhadap tindak pidana pencurian sepeda motor berdasarkan nilai yang didapatkan dari prose prediksi. Data yang digunakan untuk pemrosesan adalah data kasus pencurian dari Direktorat Kriminal Umum POLDA DIY, Satrekrim Polresta Yogyakarta, data kendaraan bermotor dari Direktorat Lalu Lintas POLDA DIY, data

penduduk dari Bagian Penduduk Biro Pemerintahan Setda DIY dan Data pengangguran dan angkatan kerja dari Dinas Tenaga Kerja dan Transmigrasi DIY serta BPS DIY. Data tersebut dalam jangka waktu Januari 2011 – Desember 2015 yang dihitung secara bulanan untuk wilayah DIY dan Kota Yogyakarta. Variabel yang digunakan dari data deret waktu tersebut adalah jumlah kasus, kendaraan, penduduk, pengangguran, serta angkatan kerja. Sehingga data yang digunakan adalah sejumlah 5 tahun x 12 bulan x 2 wilayah = 120 data per variabel, yang kemudian 120 data tersebut x 5 variabel = 600 data.

3. Prediksi Pemakaian Listrik Kelompok Tarif Menggunakan Jaringan Syaraf Tiruan dan ARIMA

   Penelitian yang dilakukan oleh Rumagit dan Azhari (2013) juga mengkombinasikan ARIMA dengan metode lainnya, yaitu Jaringan Saraf Tiruan. Kombinasi kedua metode ini dilakukan karena beberapa alasan, yaitu kesulitan dalam implementasi penggunaan model linear maupun model nonlinear pada suatu permasalahan deret waktu, juga deret waktu sangat jarang mengandung salah satunya saja (linear atau nonlinear), di mana tidak hanya model ARIMA atau jaringan syaraf masing-masing dapat memodelkan setiap kasusnya. Pada penelitian ini, data yang digunakan adalah data pemakaian listrik bulan Januari 2004 hingga bulan Desember 2010. Data bulan Januari 2004 hingga bulan Februari 2008 dijadikan sebagai data *in sample* untuk pembuatan ARIMA dan data bulan Maret 2008 hingga bulan Desember 2010 digunakan sebagai data

out sample. Input dari sistem adalah data pemakaian listrik per kelompok tarif, sedangkan output sistem adalah pemakaian listrik untuk enam periode berikutnya. Prediksi diuji pada 5 kelompok tarif yang terdiri atas kelompok tarif sosial, rumah tangga, bisnis, industri dan pemerintah. Hasil yang didapat dari perhitungan error MSE (*Mean Squared Error*) jaringan syaraf tiruan memiliki MSE yang paling besar. Sedangkan gabungan ARIMA dan JST MSE tidak berbeda jauh.MSE terkecil dari kelompok tarif sosial 0.00675, kelompok tarif rumah tangga 0.00529, kelompok tarif bisnis 0.01704, kelompok tarif industri 0.013, dan kelompok tarif pemerintah 0.0748.

4. *Comparing of ARIMA and RBFNN for Short-term Forecasting* (Perbandingan ARIMA dan RBFNN Terhadap Prediksi Jangka Pendek)

   Penelitian yang dilakukan oleh Haviluddina dan Jawahir (2015) adalah membandingkan prediksi antara ARIMA dengan metode lainnya, yaitu RBFNN (*Radial Basis Function Network*) yang merupakan bagian dari Jaringan Saraf Tiruan. Penelitian ini memprediksi kedatangan para turis di Indonesia setiap tahunnya, dari tahun 2014 hingga tahun 2018. Data yang digunakan dalam penelitian ini diambil dari halaman *web* dari BPS (http://www.bps.go.id), yang meliputi jumlah kedatangan para turis di Indonesia tahunan mulai dari tahun 1974 hingga tahun 2013. Data tersebut dianalisa dengan MATLAB, sehingga terbentuk model ARIMA dan RBFNN. Hasil perbandingan prediksi dari kedua metode tersebut adalah ARIMA menghasilkan nilai error yang lebih besar dari RBFNN,

di mana ARIMA menghasilkan nilai error sebesar 0.00722784 dan RBFNN menghasilkan nilai error sebesar 0.00098188. Perbandingan dari kedua metode tersebut membuktikan bahwa keakurasian hasil prediksi terbaik terdapat pada metode RFBNN.

5. *Optimization of ARIMA Forecasting Model Using Firefly Algorithm* (Optimalisasi Prediksi Model ARIMA dengan Algoritma Firefly)

   Penelitian yang dilakukan oleh Unggara dkk. (2019) mengikutsertakan sebuah algoritma yang dikenal dengan Algoritma *Firefly*, karena para peneliti tersebut berasumsi bahwa ARIMA memiliki kelemahan dalam menentukan model yang optimal. Optimasi yang dilakukan oleh Algoritma *Firefly* adalah mencari nilai paling rendah dari *Akaike Information Criterion* (AIC), kemudian nilai tersebut digunakan untuk menentukan model ARIMA terbaik. Penelitian ini diimplementasikan pada prediksi harga saham dan kunjungan wisatawan mancanegara ke Indonesia, sehingga data yang digunakan dalam penelitian ini adalah data saham harian Indeks Harga Saham Gabungan (IHSG) periode Januari 2013 hingga Agustus 2016 dan data kunjungan wisatawan mancanegara ke Indonesia periode Januari 1988 hingga November 2017. Hasil prediksi menunjukkan bahwa untuk data IHSG, hasil prediksi dengan model ARIMA menghasilkan RMSE (*Root Mean Square Error*) sebesar 49.72, sedangkan prediksi dengan model ARIMA Optimisasi menghasilkan RMSE sebesar 49.48. Untuk data kunjungan wisata mancanegara, hasil prediksi dengan model ARIMA menghasilkan RMSE 46088.9,

sedangkan hasil prediksi dengan ARIMA optimisasi menghasilkan RMSE 44678.4. Hasil tersebut membuktikan bahwa optimisasi model ARIMA dengan Algoritma *Firefly* menghasilkan model peramalan yang lebih baik dari pada model ARIMA tanpa Optimisasi.

6. Analisa Prediksi Jumlah Penjualan Tiket Menggunakan Metode Autoreggresive Integrated Moving Average (ARIMA) Pada PT. Charisma Rasa Sayang Holidays Medan

   Penelitian yang dilakukan oleh Chairunnisa (2015) didasari dengan adanya peningkatan terhadap jumlah penjualan di PT. Charisma Rasa Sayang Holidays Medan yang selalu menjadi bahan pokok pembicaraan yang dilakukan oleh pemimpin perusahaan itu sendiri. Pemimpin perusahaan berasumsi bahwa prediksi dapat memberikan gambaran-gambaran mengenai masa depan, sehingga kejadian yang akan datang dapat diantisipasi, misalnya jumlah penjualan tiket pada tahun 2015. Data penjualan tiket yang digunakan untuk pemrosesan adalah data penjualan sejak April 2010 hingga Agustus 2014, dan batasan penelitian yang dilakukan adalah pada musim lebaran di tahun 2015 dan penjualan yang menggunakan maskapai Singapore Airlines rute internasional. Pemrosesan data menggunakan aplikasi *SPSS,* yang terlebih dahulu mengimport data dari *Ms. Excel* ke dalam *SPSS* tersebut. Hasil dari penelitian tersebut menyimpulkan bahwa plot yang dihasilkan dari data penjualan sudah stasioner, sehingga tidak perlu dilakukan proses diferensiasi. Bukti kestasioneran lainnya adalah dari plot nilai koefisien

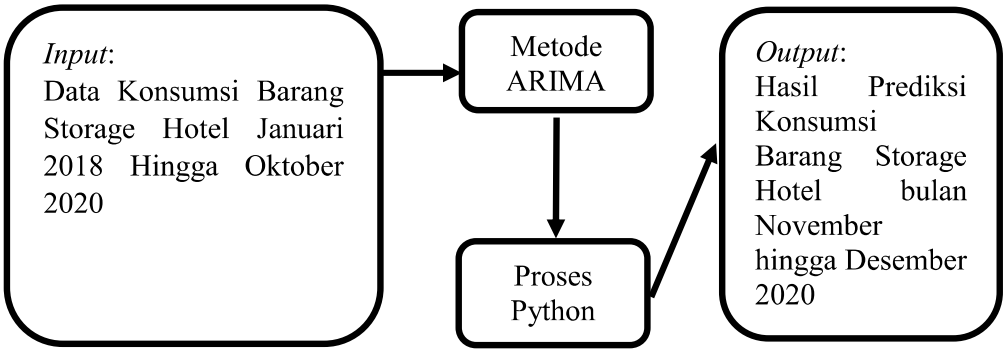
autokorelasi dan plot nilai koefisien autokorelasi data asli. Dengan memperhatikan plot nilai koefisien autokorelasi untuk mengidentifikasi proses Moving Average (MA(q) = 1), plot nilai koefisien autokorelasi parsial untuk mengidentifikasi proses Autoregressive (AR(p) = 1), sehingga diperoleh tiga model ARIMA yakni ARIMA (1, 0 , 0), ARIMA (0, 0, 1) dan ARIMA (1, 0, 1).

7. Forecasting of Sudan Inflation Rates using ARIMA Model (Prediksi Terhadap Tingkat Inflasi Sudan Dengan Model ARIMA)

   Penelitian yang dilakukan oleh (Yahia & Abdellah, 2018) didasari dengan adanya krisis ekonomi parah di negara Sudan. Data yang digunakan dalam penelitian ini adalah berasal dari Badan Pusat Statistik periode 1970 hingga 2016. Tingkat inflasi Sudan menghasilkan plot yang tidak stasioner karena bentuknya adalah trend, sehingga diperlukan logaritma natural dan mengkonversikannya ke deret waktu yang stabil. Hasil dari prediksi tersebut adalah akan ada peningkatan inflasi di tahun 2017 hingga 2026.

2.3. Kerangka Pemikiran

Berdasarkan teori-teori yang telah dijelaskan sebelumnya, kerangka pemikiran yang digunakan dalam penelitian yang dilakukan oleh peneliti adalah :



Gambar 2.1 Kerangka Pemikiran

Sumber : Data Penelitian (2020)

Pada penelitian ini, input yang digunakan adalah berupa data konsumsi barang storage hotel sejak Januari 2018 hingga November 2020. Data tersebut diproses dengan menggunakan metode *Autoregressive Integrated Moving Average* dan dibantu dengan bahasa pemrograman Python. Output dari hasil pemrosesan data tersebut adalah berupa hasil prediksi konsumsi barang storage hotel bulan November hingga Desember 2020.