BAB III
METODOLOGI PENELITIAN

3.1 Subjek dan Objek Penelitian

Subyek penelitian merujuk pada individu, objek, atau lokasi yang dianggap sebagai topik diskusi atau sebagai target dari perlakuan objek penelitian. Subjek pada penelitian ini yaitu data keadaan iklim seperti suhu, kelembapan, curah hujan dan kecepatan angin yang digunakan untuk memprediksi harga jual cabai rawit di Kota Semarang.

Objek penelitian merupakan materi, entitas, atau hal lain yang menjadi fokus, perhatian, dan subjek dari penelitian yang dilakukan. Objek pada penelitian ini yaitu model regresi *XGBoost Regression, KNN Regression* dan *Random Forest Regression* yang dibandingkan untuk mendapatkan model regresi yang akurat dalam memprediksi data

3.2 Alat dan Bahan Penelitian

Berikut merupakan alat dan bahan penelitian yang digunakan pada penelitian ini.

3.2.1 Alat Penelitian

Penelitian ini memanfaatkan instrumen penelitian yang terdiri dari komponen perangkat keras serta perangkat lunak.

a) Perangkat keras (*hardware*)

Hardware adalah elemen fisik dari suatu sistem komputer yang dapat diamati dan disentuh. Penelitian ini memanfaatkan beberapa perangkat keras berikut ini:

1) Processor Intel i5 Gen 8 (1,6 GHz)
2) Besar RAM 8GB dan storage SSD 250GB
3) Layar 14” HD 1366 x 768 resolution Acer Comfy View

b) Perangkat lunak (*software*)

Perangkat lunak atau *software* merupakan sekumpulan program atau instruksi yang dirancang untuk menjalankan tugas-tugas tertentu pada perangkat keras di komputer. Pada penelitian ini menggunakan beberapa *software* berupa:

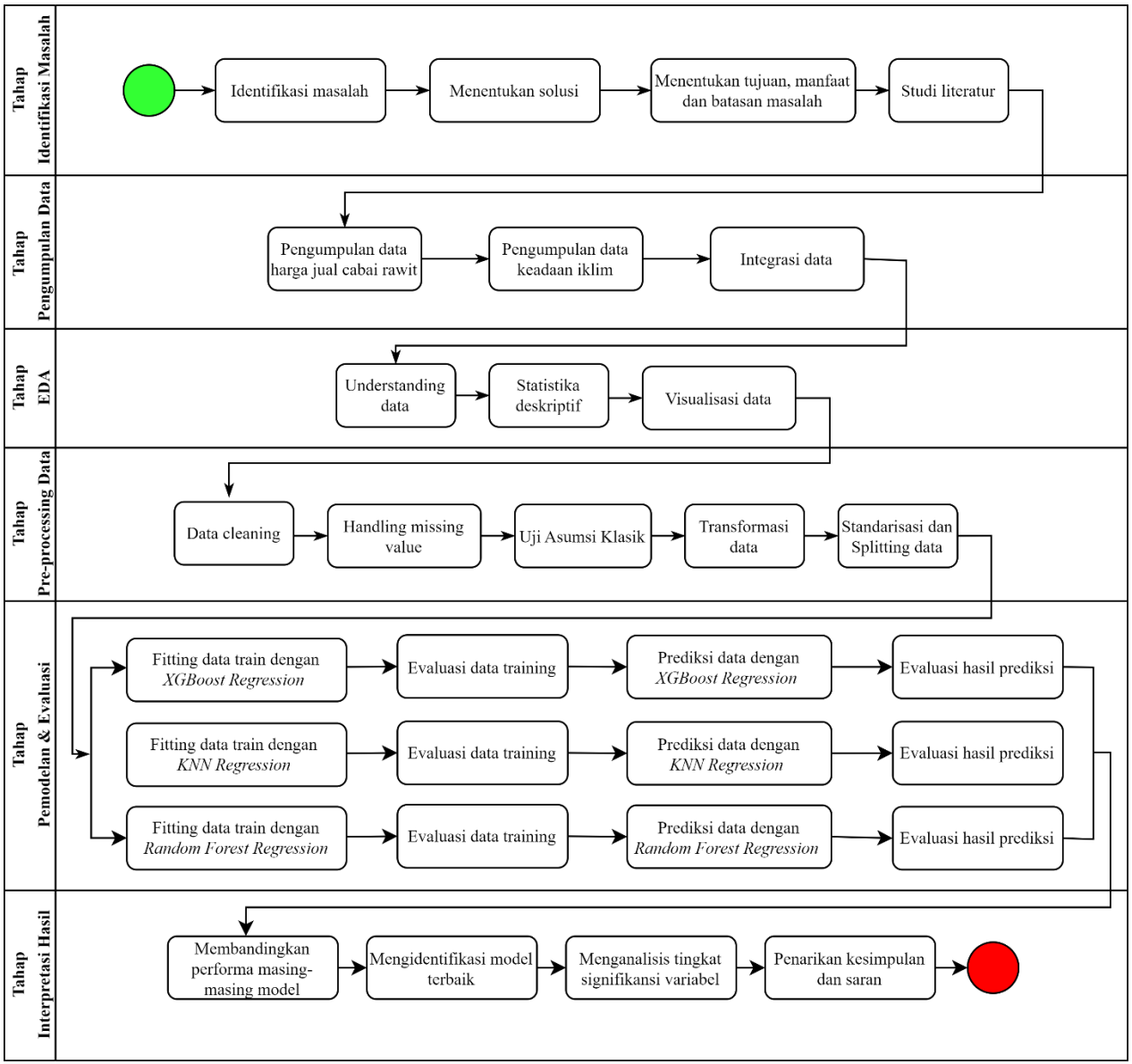
1) Sistem operasi Windows 11
2) *Chrome* sebagai peramban web
3) *Google colaboratory* sebagai *platform cloud* untuk mengeksekusi kode dari sebuah bahasa pemrograman
4) *Python* sebagai bahasa pemrograman untuk membuat model

3.2.2 Bahan Penelitian

Bahan penelitian yang digunakan dalam penelitian ini mencakup dataset keadaan iklim harian dengan variabel berupa suhu, kelembapan, curah hujan, kecepatan angin dan lama penyinaran matahari bersumber dari data *online* BMKG stasiun Meteorologi Ahmad Yani serta dataset harga jual cabai rawit harian bersumber dari *website* SiHaTi (Sistem Informasi Harga dan Produksi Komoditi) provinsi Jawa Tengah. Metode dalam mendapatkan bahan penelitian, dijelaskan dalam tahapan penelitian.

3.3 Diagram Alir Penelitian

Dalam penelitian ini menggunakan diagram alir berupa diagram *swimlane*. Diagram *swimlane* adalah representasi visual yang memperlihatkan jalannya suatu proses dengan menampilkan interaksi antara beberapa bagian yang berbeda serta pergerakan proses melalui serangkaian langkah yang berbeda. Penjelasan tentang proses dan pengelompokannya disusun dengan menempatkannya dalam beberapa baris, di mana garis-garis paralel digunakan sebagai jalur menuju kelompok subproses. Label diberikan pada jalur-jalur tersebut untuk menunjukkan struktur diagram yang disusun. Diagram alir penelitian dapat dilihat pada Gambar 3.1



Gambar 3. 1 Diagram Alir Penelitian

3.3.1 Tahap Identifikasi Masalah

Pada tahap ini dibagi menjadi beberapa bagian yaitu identifikasi masalah, menentukan solusi, menentukan tujuan, manfaat serta batasan masalah dan studi literatur. Identifikasi masalah pada penelitian ini yaitu adanya keterkaitan antara produktivitas cabai rawit terhadap keadaan iklim, sebagai contoh jika memasuki musim hujan maka terjadi penurunan jumlah panen cabai rawit. Hal ini mengakibatkan turunnya jumlah pasokan produk ke konsumen, jika hal tersebut terjadi maka penawaran tidak memenuhi permintaan sehingga berdampak pada melonjaknya harga jual.

Penentuan solusi pada penelitian ini yaitu dengan memanfaatkan teknologi *data mining* berupa analisis prediksi secara otomatis untuk mengetahui harga jual cabai rawit terhadap keadaan iklim, petani dan produsen dapat menyesuaikan pola tanam dan produksi sesuai dengan prediksi harga yang diharapkan, sehingga dapat mengoptimalkan keuntungan. Dalam penelitian juga harus menetapkan batasan agar fokus penelitian tidak berubah seiring dengan dilakukannya penelitian sampai dengan mendapatkan hasil. Beberapa batasan masalah yang ditentukan antara lain harga jual cabai rawit hanya berfokus di Kota Semarang, menggunakan data iklim dari stasiun BMKG yang berlokasi di Kota Semarang dan algoritma yang digunakan yaitu *XGBoost Regression, KNN Regression* dan *Random Forest Regression*.

Penggunaan literatur yang sesuai dengan penelitian dapat mendukung dalam mengimplementasikan teori serta konsep yang berkaitan dengan penelitian ini. Literature yang digunakan merupakan penelitian yang telah dilakukan maksimal lima tahun dari penelitian ini sehingga konsep serta wawasan yang dihasilkan masih relevan dengan perkembangan zaman.

3.3.2 Tahap Pengumpulan Data

Penelitian ini menggunakan data sekunder dengan rentang dari tahun 2016 sampai 2023. Data harga jual cabai rawit bersumber dari *website* SiHaTi (Sistem Informasi Harga dan Produksi Komoditi), dapat diakses pada link berikut: https://hargajateng.org/tabel-harga-komoditi. *Website* tersebut berisi harga jual pada beberapa komoditas seluruh wilayah di provinsi Jawa Tengah. Penelitian ini

menggunakan komoditas cabai rawit merah di Kota Semarang di mana data yang dikumpulkan merupakan data harga jual harian. Tabel 3.1 menampilkan cuplikan data harga jual cabai rawit di Kota Semarang.

Tabel 3. 1 Data Harga Jual Cabai Rawit

Tanggal	Harga Jual
01/01/2016	Rp49.600
02/01/2016	Rp49.600
03/01/2016	Rp44.400
…….	…….
29/12/2023	Rp78.000
30/12/2023	Rp75.600
31/12/2023	Rp65.400

Data keadaan iklim bersumber dari *website* data *online* BMKG (Badan Meteorologi, Klimatologi, dan Geofisika), dapat diakses pada link berikut: https://dataonline.bmkg.go.id/data_iklim. *Website* tersebut berisi data histori keadaan iklim berupa suhu, curah hujan, kecepatan angin, kelembapan dan lama penyinaran matahari. Penelitian ini menggunakan data yang berlokasi di provinsi Jawa Tengah, Kota Semarang stasiun Meteorologi Ahmad Yani. Data yang dikumpulkan merupakan data iklim harian. Tabel 3.2 menampilkan cuplikan data keadaan iklim di Kota Semarang.

Tabel 3. 2 Data Keadaan Iklim

Tanggal	Suhu	Kelembapan	Curah Hujan	Lama Penyinaran	Kecepatan Angin
01/01/2016	27	88	9,6	3,6	4
02/01/2016	27,4	87	16,7	3,4	3
03/01/2016	28,4	84	2	2,5	3
……..	……	……..	….	……..	…….
29/12/2023	28,9	84	5,6	7,9	2
30/12/2023	28,9	84	2	9	1
31/12/2023	28,9	84	0	7,7	2

Data harga jual dan keadaan iklim yang telah didapat kemudian diintegrasikan agar mempermudah dalam tahap analisis. Tabel 3.3 menampilkan cuplikan yang telah diintegrasikan.

Tabel 3. 3 Integrasi Data Keadaan Iklim Dan Harga Jual

Tanggal	Suhu	Kelembapan	Curah Hujan	Lama Penyinaran	Kecepatan Angin	Harga Jual
01/01/2016	27	88	9,6	3,6	4	Rp49.600
02/01/2016	27,4	87	16,7	3,4	3	Rp49.600
03/01/2016	28,4	84	2	2,5	3	Rp44.400
……..	……	……..	….	……..	…….	……
29/12/2023	28,9	84	5,6	7,9	2	Rp78.000
30/12/2023	28,9	84	2	9	1	Rp75.600
31/12/2023	28,9	84	0	7,7	2	Rp65.400

3.3.3 Tahap *Exploratory Data Analysis* (*EDA*)

Secara umum, *EDA* merupakan langkah awal yang penting dalam proses analisis data yang membantu untuk mendapatkan intuisi tentang data sebelum menerapkan metode analisis lebih lanjut atau membangun model. Pada tahap ini dibagi menjadi beberapa bagian yaitu *understanding* data, statistika deskriptif dan visualisasi data. *Understanding* data yaitu mengenali data secara mendalam, bagian ini mencakup spesifikasi variabel penelitian, tipe data, jumlah data duplikat, jumlah *missing value*, jumlah data *anomaly* dan *outlier*. Berikut merupakan Algoritma 3.1 untuk mengidentifikasi data yang hilang.

Algoritma 3. 1: Identifikasi *Missing Value*

```
Program   : Menghitung missing value
Deskripsi : x <- array, i <- integer
Algoritma :
missing_count <- []
For i in dataset:
      for j in dataset:
            if x[i] <- Nan
                  missing_count <- missing_count+1
      missing_values_count[column] <- missing_count
            End
      End
Output : jumlah missing value
```

Algoritma 3. 2: Identifikasi Jumlah Data *Anomaly*

```
Program   : Menghitung nilai 8888
Deskripsi : x <- array, i <- integer
Algoritma :
Nilai_anomaly <- []
For i in dataset:
      If x[i] <- 8888, then:
            Nilai_ anomaly <-  Nilai_ anomaly + 1
      End
Output : Jumlah data 8888
```

Bagian dari *EDA* lainnya yaitu statistika deskriptif. Berikut merupakan Algoritma 3.3 untuk mengidentifikasi statistika deskriptif data.

Algoritma 3. 3: Statistika Deskriptif Data

```
Program   : Menghitung statistika deskrpitif
Deskripsi : x <- array, i <- integer
Algoritma :
n <-  length(x)
Sum <-  []
Sorted_data <-  sort(x)
Min_value <-  data[i]
Max_value <-  data[i]
For i in dataset:
      Sum <-  sum+data[i]
      If data[i] < Min_value:
            Min_value <-  data[i]
      End
      If data[i] > Max_value:
            Max_value <-  data[i]
      End

Mean <-  sum/n
If n % 2 <-  0:
      Median <-  (Sorted_data[n/2 – 1] + Sorted_data[n/2])/2
Else:
      Median <-  Sorted_data[(n-1)/2]
End
Output : max, min, mean, median
```

dasar tentang suatu dataset. Adapun beberapa aspek dalam statistika deskriptif seperti rata-rata, *median, range,* standar deviasi dan kuartil.

Visualisasi data dalam *EDA* berguna untuk memberikan pemahaman yang lebih baik tentang struktur, pola, dan karakteristik kumpulan data. Jenis visualisasi yang digunakan antara lain *heatmap correlation*, bertujuan untuk melihat besarnya korelasi antar variabel satu dengan yang lain. Berikut merupakan Algoritma 3.4 untuk visualisasi *Heatmap Correlation*.

Algoritma 3. 4: Korelasi Fitur

```
Program   : Korelasi Fitur
Deskripsi : variabel1 <- array, variabel2 <- array,  i <- integer
Algoritma :
Mean_Variabel1 <- mean(variabel1)
Mean_Variabel2 <- mean(variabel2)
N <- length(variabel1)
Covariance = 0
For i in dataset:
      Covariance <- Covariance+(variabel1[i]- Mean_Variabel1)*
                    (variabel2[i]- Mean_Variabel2)
      Covariance <- Covariance/(n-1)
End
Stdev_vaiabel1 <- STD_DEV(variabel1)
Stdev_vaiabel2 <- STD_DEV(variabel2)
Corr <- Covariance/( Stdev_vaiabel1* Stdev_vaiabel1)
Output : Korelasi antar variabel
```

Boxplot bertujuan untuk melihat sebaran data yang *outlier* serta bentuk distribusi data. Histogram bertujuan untuk melihat apakah data tersebut berdistribusi normal atau tidak serta melihat keruncingan dari sebaran data. Diagram garis bertujuan untuk melihat pola dataset secara *time series*. Beberapa visualisasi tersebut juga dapat digunakan untuk mendeteksi adanya *anomaly* dalam dataset.

3.3.4 Tahap *Pre-processing* Data

Tahap ini dilakukan untuk mempersiapkan data sebelum masuk ke dalam pemodelan dengan tujuan untuk menyesuaikan data sesuai dengan karakteristik dari masing-masing algoritma yang digunakan dalam pemodelan. Pada tahap ini dibagi menjadi *data cleaning*, penanganan *missing value*, transformasi data dan standarisasi serta *splitting* data. *Data cleaning* dilakukan untuk memastikan data konsisten, tidak ada kesalahan format data, tidak ada atribut tambahan serta menghapus entri maupun baris yang duplikat. Berikut merupakan Algoritma 3.5 untuk melakukan *data cleaning.*

Algoritma 3. 5: Mengganti Nilai Tertentu menjadi NaN

```
Program   : Mengganti nilai 8888 menjadi NaN
Deksripsi : x <- array,  i <- integer
Algoritma :
Value <- []
rows <- length(dataset)
      If rows <- 0:
            Return dataset
      End
columns <- length(data[i])
For i from 0 to rows-1:
      For j from 0 to rows-1:
            If x[i][j] <- Value:
                  Dataset[i][j] <- Nan
            End
      End
Output : replace 8888 menjadi Nan
```

Penanganan *missing value* bertujuan untuk meningkatkan kualitas dan keakuratan dalam analisis data. Berikut merupakan Algoritma 3.6 untuk melakukan penanganan *missing value.* Pengisian *missing value* menggunakan teknik pengisian dengan nilai statistik. Pengisian dengan nilai statistik menggunakan nilai *mean*, *median* maupun modus sesuai dengan karakteristik variabel yang memiliki *missing value*.

Algoritma 3. 6: Penanganan *Missing Value*

```
Program   : Imputasi missing value dengan mean
Deskripsi : x <- array,  i <- integer
For i in dataset:
      Sum <- []
      Count <- []
            For j in columns:
                  If j[i] not Nan:
                        Sum <- sum+j
                        Count <- count+1
                  End
            End
            Mean <- sum/count
            For j in columns:
                  If j[i] <- Nan:
                        j <- mean
                  End
            End
Output = Mengganti Nan menjadi mean
```

Setelah data dipastikan tidak memiliki *missing value*, maka selanjutnya yaitu uji asumsi klasik data yang bertujuan untuk memastikan bahwa model regresi yang dibangun memenuhi asumsi-asumsi kalsik yang diperlukan, adapun jenis ujinya yaitu uji normalitas, uji homoskedestisitas, uji multikolinearitas, uji autokorelasi dan uji linearitas. Apabila dari semua uji asumsi klasik tersebut ada yang tidak terpenuhi, maka dapat dipastikan pemodelan dilakukan menggunakan regresi non-*linear*,

a) Uji Normalitas

Uji normalitas merupakan uji statistik yang digunakan untuk menguji asumsi distribusi normal pada data. Data populasi dikatakan berdistribusi normal jika nilai rata-rata mengumpul di bagian tengah, nilai *mode* dan mediannya berada pada batas kewajaran tertentu. Selanjutnya penarikan kesimpulan dilakukan dengan merumuskan hipotesis yang didasari pada tingkat kesalahan dan statistik hitung. Selain itu uji hipotesis juga dapat dilakukan dengan membandingkan *p-value* dengan tingkat kesalahan yang sesuai. Metode uji *Kolmogorov-Smirnov* digunakan untuk mengetahui apakah suatu data mengikuti suatu distribusi tertentu ketika *mean* dan variansinya diketahui, statsitika uji *Kolmogorov-Smirnov D* didefinisikan pada persamaan 3.1 dan persamaan 3.2:

$$D^{+} = \max\left(\left|\frac{i}{n} - F(x_i)\right|\right) \qquad (3.1)$$

$$D^{-} = \max\left(\left|F(x_i) - \frac{i-1}{n}\right|\right), \qquad (3.2)$$

dengan $F(x_i)$ merupakan fungsi kumulatif distribusi normal, *n* menyatakan jumlah sampel dan *i* merupakan *indeks* pada data. Adapun pengambilan keputusan dapat dilihat pada daerah kritis, apabila nilai *D* lebih besar dari nilai *D* tabel ($D > D_{tab}$), maka tolak H_0 dan terima H_1 dimana H_0 menyatakan bahwa data memiliki distribusi normal [43].

b) Uji Heteroskedastisitas

Uji ini bertujuan menguji apakah model regresi terjadi ketidaksamaan varians dari residual atau pengamatan ke pengamatan lain. Jika varians dari residual suatu pengamatan ke pengamatan lain tetap disebut homokedastisitas, sedangkan untuk varians yang berbeda disebut heteroskedastisitas. Uji *glejser* adalah uji statistik yang paling umum digunakan, uji *glejser* mengusulkan untuk meregres nilai *absolut* residual terhadap variabel independen. Uji *glejser* didefinisikan pada persamaan 3.3:

$$|U_t| = a + Bx_t + v_t, \qquad (3.3)$$

dengan $|U_t|$ merupakan nilai *absolut* residual, a merupakan *intercept*, *B* merupakan koefisien dari variabel independen, x_t merupakan variabel independen dan $v_t$merupakan *error* yang dihasilkan uji ini. Model regresi dikatakan tidak mengandung heterokedastisitas jika probabilitas signifikansinya di atas tingkat kepercayaan 5% atau > 0,05 dan sebaliknya [44].

c) Uji Autokorelasi

Tujuan dari uji ini yaitu untuk mengetahui apakah dalam sebuah model regresi *linear* ada korelasi antara kesalahan pada periode *t* dengan kesalahan pada periode sebelumnya. Jika terjadi korelasi, maka ada problem autokorelasi. Salah satu cara untuk mengetahui ada tidaknya autokorelasi pada model regresi adalah dengan melakukan Uji *Durbin Watson* (*DW*) yang didefinisikan pada persamaan 3.4:

$$DW = \frac{\sum_{t=2}^{T}(e_t - e_{t-1})^2}{\sum_{t=1}^{T} e_t^2}, \qquad (3.4)$$

dengan e_t adalah angka sisa, *T* adalah banyaknya observasi percobaan dan *t-1* merupakan periode sebelumnya. Pengambilan keputusan yaitu jika nilai *DW*

kurang dari batas bawah (*du*) maka terjadi autokorelasi positif dan lebih dari batas atas (*dl*) maka terjadi autokorelasi *negative* [44].

d) Uji Linearitas

Uji linearitas merupakan salah satu pengujian yang harus dilakukan sebelum menentukan model regresi, di mana pengujian ini bertujuan untuk melihat terdapat atau tidaknya hubungan yang linear dari seluruh sebaran data. Linearitas terpenuhi dengan asumsi apabila *plot* antara nilai residual terstandarisasi dengan nilai prediksi terstandarisasi tidak membentuk suatu pola tertentu atau random. Uji F dilakukan untuk melihat pengaruh dari seluruh variabel bebas secara bersama-sama terhadap variabel terikat, uji ini didefinisikan pada persamaan 3.5:

$$F = \frac{\frac{(RSS_0 - RSS)}{(p-1)}}{\frac{RSS}{(n-p)}}, \qquad (3.5)$$

dengan RSS_0 adalah *Residual Sum of Square* dari model yang hanya dimiliki *intercept*, *RSS* adalah *Residual Sum of Square* dari keseluruhan model, *p* merupakan jumlah parameter dan *n* merupakan jumlah observasi. Tingakatan yang digunakan sebesar 0.5 atau 5%, jika nilai signifikan F < 0.05 maka dapat diartikan bahwa variabel independent secara simultan mempengaruhi variabel dependen ataupun sebaliknya [15][44].

e) Uji Multikolinearitas

Multikolinearitas merupakan keadaan dimana terjadi hubungan linear yang sempurna atau mendekati antar variabel independen dalam model regresi. Suatu model regresi dikatakan mengalami multikolinearitas jika ada fungsi linear yang sempurna pada beberapa atau semua independen variabel dalam fungsi linear. Gejala adanya multikoliniearitas antara lain dengan melihat nilai *Variance Inflation Factor* (*VIF*) dan *Tolerance* yang didefinisikan pada persamaan 3.6:

$$VIF_i = \frac{1}{1-R_i^2}, \qquad (3.6)$$

dengan R_i^2 adalah koefisien determinasi dari regresi variabel independen *i* terhadap semua variabel independen lainnya dalam model. Jika nilai *VIF* < 10

dan *Tolerance* > 0,1, maka dinyatakan tidak terjadi multikolinearitas, begitupun sebaliknya [15].

Setelah uji asumsi kalsik dilakukan, tahapan selanjutnya yaitu transformasi data . Transformasi data digunakan untuk mengubah nilai-nilai dalam data untuk memenuhi persyaratan tertentu. Pemilihan metode transformasi data tergantung pada tujuan analisis data. Penelitian ini menggunakan transformasi *BoxCox* yang umum digunakan untuk mengubah data yang tidak berdistribusi normal menjadi mendekati normal dan menangani data yang memiliki banyak *outlier*. Berikut merupakan Algoritma 3.7 untuk melakukan transformasi data.

Algoritma 3. 7: Transformasi *BoxCox*

```
Program   : Transformasi BoxCox
Deskripsi : x <- array,  i <- integer
Algoritma :
Function boxcox_trasnform(dataset):
      lambda <- best_lambda(dataset)
      if abs(lambda) < epsilon:
            transformed_data <- log_transform(dataset)
      else:
            transformed_data <- apply_boxcox(dataset, lambda)
      End
function best_lambda(dataset):
      normality_scores <- []
      for each lambda_value in lambda_values:
            transformed_data <- apply_boxcox(dataset, lambda)
            normality_scores <- compute_normality_score
                                (transformed_data)
            normality_scores.append(lambda_value, normality_scores)
            End
      lambda <- select_best(normality_scores)
      End
function apply_boxcox(data, lambda_value)
      if lambda_value <- 0:
            transformed_data <- log_transform(dataset)
      else:
            transformed_data <- (data**lambda_value-1)/lambda_value
      End
output : transformasi pada fitur numerik
```

Adapun langkah-langkah dalam melakukan transformasi ini yaitu memastikan tidak ada data nol atau negatif, pilih variabel x yang ditransformasi, tentukan nilai *lambda* (λ) dengan rentang nilai dari -5 sampai 5 menggunakan metode *likelihood* atau menggunakan *library boxcox* di *Python* dan terakhir yaitu transformasi nilai-nilai yang ada di variabel x menggunakan persamaan (2.1).

Setelah melakukan transformasi, tahap selanjutnya yaitu melakukan standarisasi data menggunakan metode *StandardScaler*. Berikut merupakan Algoritma 3.8 untuk melakukan standarisasi data.

Algoritma 3. 8: Standarisasi *StandarScaler*

```
Program   : Standarisasi StandarScaler
Deskripsi : x <- array,  i <- integer
Algoritma :
Mean <- []
Stdv <- []
Num_feature <-length(dataset[0])
for i from dataset:
      Feature_sum <- 0
      Sample <- length(dataset)
      For j in dataset (sample-1):
            Feature_sum <- Feature_sum+dataset[j][i]
      End
      Feature_mean <- Feature_sum/Sample
      Append Feature_mean to Mean
End
for i from dataset:
      Sum_of_square <- 0
      Sample <- lengrh(dtaset)
      For j from dataset:
            Sum_of_square <- Sum_of_square + (dataset[j][i]-
                             mean[i])^2
            End
      feature_std_dev <- SQRT(Sum_of_square/Sample)
      append feature_std_dev to Stdv
      End
for i from dataset:
      for j from dataset:
            dataset[j][i] <- (dataset[j][i]-mean[i]Stdev[i]
            End
      End
output : standarisasi data pada fitur numerik
```

Tahap akhir pada *pre-processing* data yaitu *Split* dataset. Tahapan ini merupakan proses membagi dataset menjadi dua bagian, yaitu data latih dan data uji. *Split* dataset membantu mengukur sejauh mana model dapat menggeneralisasi dari data latih ke data baru yaitu data uji. Dengan melakukan *split* pula, performa model dapat diukur untuk melihat sebaik apa model memprediksi data baru dalam hal ini yaitu data uji.

3.3.5 Tahap Pemodelan dan Evaluasi

Tahap ini dibagi menjadi beberapa bagian yaitu *fitting* data latih dengan masing-masing model, evaluasi dari hasil *fitting* data latih, prediksi data dengan

masing-masing model dan evaluasi dari hasil prediksi. Penelitian ini menggunakan tiga model yaitu *XGBoost Regression*, *KNN Regression* dan *Random Forest Regression*. Uji coba pada algoritma *XGBoost Regression* menggunakan beberapa variasi nilai-nilai pada parameter. Tabel 3.4 merupakan variasi beberapa kombinasi nilai parameter yang dilatih pada dataset.

Tabel 3. 4 Kombinasi Nilai Pada Parameter *XGBoost Regression*

Parameter	Kombinasi 1	Kombinasi 2	Kombinasi 3
objective	*reg:squarederror*	*reg:squarederror*	*reg:squarederror*
n_estimators	50	100	150
max_depth	3	4	5
learning_rate	0.01	0.05	0.1
subsample	0.5	0.6	0.8
colsample_bytree	0.5	0.6	0.8
gamma	0	0.1	0.2

Algoritma 3. 9: Algortima *XGBoost Regression*

```
Program   : Algortima XGBoost Regression
Deskripsi : x,y <- array, parameter
Algoritma :
num_trees <- []
FUNCTION train_model(x, y, parameters):
    FOR each tree in range(num_trees):
        #Residual
        IF tree <- 0:
            predictions <- initial_prediction
        ELSE:
            predictions <- predictions - learning_rate * gradient
        End
    End
        #Hitung gradient dan hessian
        gradient <- compute_gradient(data, predictions, targets)
        hessian <- compute_hessian(data, predictions)
        End
        #Bangun pohon
        tree <- build_tree(data, gradient, hessian, parameters)
        End
        #Perbaiki prediksi dari pohon baru
        predictions <-predictions + learning_rate * predict(tree,
                     data)
        End
        #Membuat pohon baru pada model
        model['trees'].append(tree)
    End
FUNCTION predict(model, x):
    predictions <- 0
    FOR each tree in model['trees']:
        predictions <-predictions + learning_rate * predict(tree,
                      data)
Output : Model xgboost yang dapat melatih kasus regresi
```

Uji coba pada algoritma *KNN Regression* menggunakan beberapa variasi nilai-nilai pada parameter. Tabel 3.5 merupakan variasi beberapa kombinasi nilai parameter yang dilatih pada dataset.

Tabel 3. 5 Kombinasi Nilai Pada Parameter *KNN Regression*

Parameter	Kombinasi 1	Kombinasi 2	Kombinasi 3
n_neighbors	4	5	6
weights	*uniform*	*uniform*	*uniform*
algorithm	*auto*	*auto*	*auto*
leaf_size	10	20	30
p	1	2	*np.inf*

Algoritma 3. 10: Algortima *KNN Regression*

```
Program   : Algortima KNN Regression
Deskripsi : x,y <- array,  k <- integer
Algoritma :
FUNCTION knn_regression(x, y, k):
    #Hitung jarak
    FOR each instance in training_set:
        distance <- calculate_distance(k, y)
    End
    #Urutkan instance dalam training set berdasarkan jarak
    Sort x by distance
    End
    #Pilih k tetangga terdekat
    neighbors <- select_k_nearest_neighbors(x, k)
    End
    # Hitung nilai prediksi
    prediction <- calculate_regression(neighbors)
    End

FUNCTION calculate_distance(instance1, instance2):
    #Hitung jarak antara dua instance menggunakan Euclidean
    distance <- sqrt(sum((instance1 - instance2)^2))
    End
Output : Model knn yang dapat melatih kasus regresi
```

Uji coba pada algoritma *Random Forest Regression* menggunakan beberapa variasi nilai-nilai pada parameter. Tabel 3.6 merupakan variasi beberapa kombinasi nilai parameter yang dilatih pada dataset.

Tabel 3. 6 Kombinasi Nilai Pada Parameter *Random Forest Regression*

Parameter	Kombinasi 1	Kombinasi 2	Kombinasi 3
max_depth	2	5	10
max_features	*auto*	*auto*	*auto*
max_leaf_nodes	5	6	7
n_estimators	50	100	150

Algoritma 3. 11: Algortima *Random Forest Regression*

```
Program   : Algortima Random Forest Regression
Deskripsi : x,y <- array,  parameter
Algoritma :
FUNCTION random_forest_regression(x, num_trees):
    #Membuat pohon-pohon keputusan
    FOR i FROM 1 TO num_trees:
        #Acak sample
        bootstrap_sample <- random_sampling_with_replacement
                            (dataset)
        End
        #Acak subset fitur
        random_features <- random_subset_of_features(x)

        #Bangun pohon-pohon keputusan
        decision_tree <- build_decision_tree(bootstrap_sample,
                         random_features)

        # Tambahkan pohon-pohon baru ke model
        forest.append(decision_tree)
    End
FUNCTION predict_random_forest(forest, sample):
    predictions <- []

    #Prediksi setiap pohon dalam model
    FOR tree IN forest:
        # Lakukan prediksi dengan pohon
        prediction <- predict_with_decision_tree(tree, sample)

        # Tambahkan hasil prediksi ke daftar
        predictions.append(prediction)
    End
    # Return agregasi hasil prediksi dari setiap pohon
    RETURN average(predictions)

Output : Model random forest yang dapat melatih kasus regresi
```

Setelah data latih di *fitting* dengan masing-masing model, maka hasil dari pemodelan tersebut dievaluasi. Matriks evaluasi menggunakan *MAE, MAPE* dan *R2-score* untuk menilai seberapa baik model dalam memprediksi data dan *error* yang dihasilkan masing-masing model. Model yang dihasilkan dari *fitting* data latih, kemudian digunakan kembali untuk memprediksi data uji. Hasil dari prediksi juga dievaluasi menggunakan *MAE, MAPE* dan *R2-score.* Hasil evaluasi dari data latih dan data uji digunakan untuk melihat apakah model mengalami *overfitting.* Apabila evaluasi antara data latih dan data uji memiliki *merge* yang besar, maka model tidak dapat melakukan generalisasi dengan baik dan model harus di *fitting* kembali. Algoritma 3.12 merupakan algoritma dari matriks evaluasi.

Algoritma 3. 12: Matriks Evaluasi

```
Program   : Matriks evaluasi
Deskripsi : y_true, y_pred <- array
Algoritma :
Menghitung MAE:
mae <-  np.mean(np.abs(y_true - y_pred))

Menghitung MAPE:
mape <-  np.mean(np.abs((y_true - y_pred) / y_true)) * 100

Menghitung R2-Score (Coefficient of Determination):
mean_y_true = np.mean(y_true)
ss_tot <-  np.sum((y_true - mean_y_true)**2)
ss_res <-  np.sum((y_true - y_pred)**2)
r2 <-  1 - (ss_res / ss_tot)

Output: Matriks evaluasi MAE, MAPE dan R2-Score
```

3.3.6 Tahap Interpretasi Hasil

Pada tahap ini membahas hasil yang telah didapatkan selama penelitian dilakukan. Tahap ini juga menjawab pertanyaan penelitian yang telah ditentukan sebelumnya. Adapun pembagian pada tahap ini meliputi perbandingan performa masing-masing model, identifikasi model terbaik dan analisis tingkat signifikansi antar variabel. Perbandingan performa masing-masing model menggunakan matriks evaluasi yang sudah ditentukan sebelumnya. Evaluasi yang dibandingkan merupakan hasil dari prediksi masing-masing model. Tabel 3.9 merupakan ilustrasi perbandingan antar model.

Tabel 3. 7 Perbandingan Performa Antar Model

Model	Matriks Evaluasi		
	MAE	*MAPE*	*R2-Score*
XGBoost Regression	……..	……..	……..
KNN Regression	……..	……..	……..
Random Forest Regression	……..	……..	……..

Identifikasi model terbaik jika memenuhi persyaratan yaitu memiliki *MAE* terendah, *MAPE* terendah dan *R2-Score* terbesar. Setelah model terbaik didapat, analisis selanjutnya yaitu pengaruh signifikan dan kontribusi antar variabel x terhadap variabel y. Pengaruh signifikan variabel x dihasilkan dari model yang memiliki performa terbaik. Untuk mengetahui kontribusi setiap variabel x terhadap variabel harga jual menggunakan metode *SHAP* dengan menghitung nilai *Shapely*

pada masing-masing variabel. Tujuan dari analisis tersebut untuk mengetahui variabel iklim apa yang berpengaruh signifikan terhadap harga jual cabai rawit. Algoritma 3.13 merupakan algoritma dari *SHAP* dan nilai *Shapely*. Setelah semua interpretasi hasil dilakukan, maka selanjutnya masuk ke dalam penarikan kesimpulan dan saran penelitian. Setelah semua kegiatan penelitian telah dilakukan sesuai dengan tahapan yang sudah ditentukan, maka penelitian selesai dilaksanakan.

Algoritma 3. 13: Nilai *SHAP*

```
Program   : Menghitung nilai SHAP
Deskripsi : x, y <- array
Algoritma :
FUNCTION calculate_SHAP_values(model, x, baseline):
    shap_values <- []
    FOR each instance in dataset:
        # Hitung nilai baseline
        baseline_prediction <- model.predict(baseline)

        # Hitung prediksi model untuk instance saat ini
        current_prediction <-  model.predict(instance)

        # Inisialisasi array untuk menyimpan nilai SHAP
        instance_shap_values <-  []

        FOR each feature in instance:
            # Buat instance dengan feature yang dihilangkan
            instance_without_feature <- instance.copy()
            instance_without_feature[feature] <-
            baseline[feature]

            # Hitung prediksi model untuk instance tanpa feature
            prediction_without_feature <-
            model.predict(instance_without_feature)

            # Hitung SHAP value untuk feature saat ini
            shap_value <- current_prediction –
            prediction_without_feature + baseline_prediction

            # Tambahkan SHAP value ke dalam array
            instance_shap_values.append(shap_value)
         End
     End
Output : Nilai SHAP pada masing-masing fitur
```