

# LAPORAN TUGAS BESAR PEMROGRAMAN PYTHON

\$ 🥑 AVOCADO PRICE 🥑 \$

Disusun sebagai syarat mata kuliah Pemrograman Python

pada Program Studi S1 Teknik Telekomunikasi

Oleh

[ Kelompok B ]

MUHAMMAD AGUS ATHARIQ      1101220267

RAIHAN RAMADHAN SETIAWAN      1101223012

HUSNA FIRYAL AZ-ZAHRA      1101223038

JOSEPHINE MONICA      1101223063



FAKULTAS TEKNIK ELEKTRO

UNIVERSITAS TELKOM

BANDUNG

2023

## BAB I

### PENDAHULUAN

#### 1.1 Deskripsi Tugas

##### Context

It is a well-known fact that Millenials LOVE Avocado Toast. It's also a well-known fact that all Millenials live in their parent's basements.

Clearly, they aren't buying home because they are buying too much Avocado Toast!

But maybe there's hope... if a Millenial could find a city with cheap avocados, they could live out the Millenial American Dream.

##### Content

This data was downloaded from the Hass Avocado Board website in May of 2018 & compiled into a single CSV. Here's how the Hass Avocado Board describes the data on their website:

The table below represents weekly 2018 retail scan data for National retail volume (units) and price. Retail scan data comes directly from retailers' cash registers based on actual retail sales of Hass avocados. Starting in 2013, the table below reflects an expanded, multi-outlet retail data set. Multi-outlet reporting includes an aggregation of the following channels: grocery, mass, club, drug, dollar, and military. The Average Price (of avocados) in the table reflects a per unit (per avocado) cost, even when multiple units (avocados) are sold in bags. The Product Lookup codes (PLU's) in the table are only for Hass avocados. Other varieties of avocados (e.g. greenskins) are not included in this table.

Some relevant columns in the dataset:

- Date - The date of the observation
- AveragePrice - the average price of a single avocado
- type - conventional or organic
- year - the year
- Region - the city or region of the observation
- Total Volume - Total number of avocados sold
- 4046 - Total number of avocados with PLU 4046 sold
- 4225 - Total number of avocados with PLU 4225 sold
- 4770 - Total number of avocados with PLU 4770 sold

## 1.2 Analisis Tugas

Dalam data yang dilampirkan dalam deskripsi tugas, diberikan sejumlah data yang berasal dari situs web *Hass Avocado Board*, dan dikompilasi ke dalam file CSV. Di dalamnya terdapat data penjualan alpukat jenis *hass* yang mencakup:

- Tanggal (tanggal observasi)
- Harga Rata-rata (harga rata-rata satuan alpukat)
- Tipe (alpukat konvensional atau alpukat organic)
- Tahun
- Wilayah (kota atau daerah observasi)
- Total Volume (volume alpukat yang terjual)
- PLU 4046
- PLU 4225
- PLU 4770

Kode PLU (*Price Look-Up*) adalah kode numerik yang digunakan dalam industri pertanian untuk mengidentifikasi dan membedakan jenis buah atau sayuran tertentu. Khusus untuk konteks data tentang alpukat yang disebutkan diatas, kode PLU digunakan untuk mengidentifikasi varietas tertentu dari alpukat Hass. Oleh karena itu, kode PLU pada data tersebut dapat digunakan untuk membedakan alpukat Hass dari variasi alpukat lainnya, seperti alpukat jenis greenskins yang tidak termasuk dalam tabel data tersebut.

## BAB II

### PEMBAHASAN

#### 2.1 Library yang Digunakan

Library Python adalah kumpulan modul atau kode program yang telah ditulis sebelumnya dan dapat digunakan kembali untuk mempermudah proses pengembangan aplikasi. Dalam ekosistem Python yang kaya dan dinamis, terdapat ribuan library yang tersedia untuk memenuhi berbagai kebutuhan pemrogram. Library menyediakan sekumpulan alat dan fungsi yang dapat diimpor ke dalam program Python.

Salah satu keunggulan utama dari penggunaan library Python adalah memungkinkan pemrogram untuk menghemat waktu dan upaya dengan menggunakan fungsi-fungsi yang sudah ada, daripada harus menulis ulang kode dari awal. Dengan menggunakan library, pengembang dapat fokus pada masalah khusus yang ingin mereka selesaikan, sementara kerangka kerja umum dan fungsi dasar telah diatasi oleh library.

Pada penugasan mata kuliah Pemrograman Python kali ini kami menggunakan beberapa library, diantaranya:

- Numpy
- Pandas
- Matplotlib Pyplot
- Seaborn
- Datetime
- Plotly

#### 2.2 Numpy

NumPy adalah paket dasar untuk komputasi ilmiah dengan Python. Mendukung objek array multidimensi, berbagai objek turunan (seperti array dan matriks), dan berbagai rutinitas untuk operasi cepat pada array seperti matematika, logika, manipulasi bentuk, pengurutan, seleksi, I/O, Fourier diskrit, dll. Pustaka Python yang menyediakan Transformasi, aljabar linier dasar, operasi statistik dasar, simulasi acak, dan banyak lagi. Library Pandas memiliki keunggulan, yaitu:

- **Melakukan perhitungan array**, Penggunaan Python untuk pengolahan data tersebar luas. Kita bisa menggunakan urutan Python standar, tetapi NumPy Python menyederhanakan komputasi ilmiah. Kita bisa lebih mudah

menggunakan Numpy untuk perhitungan ilmiah tanpa harus menulis rumus dengan tangan.

- **Komputasi berorientasi array**, Sifat komputasi yang berorientasi pada array memudahkan penghitungan manipulasi data menggunakan Numpy Python.
- **Menyediakan Fungsi Bawaan untuk Aljabar Linier dan Pembuatan Bilangan Acak**, Selain kemampuan untuk menggunakan metode reshape(), NumPy Python juga menyediakan fungsi bawaan lainnya. Fungsi yang memudahkan pengolahan data antara lain fungsi aljabar linier dan fungsi pembangkitan bilangan acak. Dapat menghemat waktu dan membuat pemrosesan data menjadi lebih efisien.

### 2.3 Pandas

Pandas Library adalah sebuah perpustakaan open source untuk bahasa pemrograman Python yang banyak digunakan untuk pengolahan data, mulai dari pembersihan data, manipulasi data, hingga analisis data. Pandas library didasarkan pada dua pustaka inti Python: Matplotlib untuk visualisasi data dan NumPy untuk operasi matematika. Library Pandas memiliki keunggulan, yaitu:

- **Memiliki banyak fitur**, Pustaka Pandas menyediakan serangkaian fitur yang luas dan penting untuk menganalisis data dengan mudah. Dengan menggunakan perpustakaan Pandas untuk melakukan berbagai tugas, termasuk: Contoh: memfilter data berdasarkan kriteria atau kelompok tertentu, memisahkan data berdasarkan preferensi, dll.
- **Mempersingkat pengambilan data**, Library Pandas dapat menulis lebih sedikit dan mengerjakan lebih banyak sekaligus. Misalnya, menjalankannya dengan Python tanpa pustaka akan memakan waktu lebih lama karena ada lebih banyak baris yang harus diurai. Sebaliknya, jika menggunakan Pandas Library, Hanya perlu menjalankan beberapa baris saja. Hal ini dapat mempersingkat proses pengolahan data. Penghematan waktu ini memungkinkan untuk lebih fokus pada algoritma analisis data.
- **Menyediakan representasi data**, Pustaka Pandas menyediakan bentuk representasi data yang sangat optimal. Ini membantu dalam menganalisis dan

memahami data secara lebih dalam. Selain itu, representasi data yang lebih sederhana menghasilkan hasil yang lebih baik dalam proyek terkait data.

#### **2.4 Matplotlib Pyplot**

Matplotlib.pyplot adalah library yang digunakan dalam bahasa pemrograman python untuk melakukan visualisasi data secara 2D maupun 3D. Secara umum matplotlib.pyplot dapat disingkat menjadi plt. Visualisasi data merupakan aspek kunci dalam analisis karena dapat membantu memahami tren dan pola dalam data yang dikerjakan. Matplotlib ini memungkinkan kita untuk mengubah data mentah menjadi informasi yang menarik melalui berbagai bentuk grafik dan diagram. Kelebihan dari Matplotlib adalah memberikan kemudahan dalam penggunaan. Hanya dengan menggunakan beberapa baris kode sederhana, kamu bisa menciptakan visualisasi yang kompleks.

#### **2.5 Seaborn**

Seaborn adalah library visualisasi data dalam bahasa pemrograman python Python yang menyediakan high-level interface (antarmuka tingkat tinggi) untuk menggambar grafik statistik yang informatif. Seaborn dibangun di atas Matplotlib dengan memanfaatkannya sebagai dasar. Seaborn memudahkan dalam memvisualisasikan data dengan berbagai jenis plot untuk analisis statistik seperti distribusi, kategorikal, matriks, dan banyak lagi. Seaborn bekerja dengan sangat baik dengan DataFrame Pandas sehingga memudahkan pengguna untuk memvisualisasikan data langsung dari DataFrame.

#### **2.6 Datetime**

Dalam bahasa pemrograman Python, datetime adalah modul tunggal, artinya ini bukan dua tipe data yang terpisah. Kita dapat mengimpor modul datetime ini sedemikian rupa sehingga dapat berfungsi dengan tanggal dan waktu. Pengembang tidak perlu menginstalnya secara terpisah. Itu ada di sana. Dengan modul datetime, kita dapat bekerja dengan kelas berbeda yang bekerja sempurna dengan tanggal dan waktu. Dalam kelas-kelas ini, Kita bisa mendapatkan berbagai fungsi yang berhubungan dengan tanggal, waktu, dan interval waktu yang berbeda.

Kelebihan dari library ini antara lain:

- Modul ini menyediakan dukungan yang komprehensif untuk manipulasi tanggal dan waktu, termasuk operasi aritmatika, konversi, dan format.
- Datetime Python dirancang untuk efisien, sehingga cocok untuk aplikasi yang membutuhkan kinerja tinggi.
- Datetime Python terintegrasi dengan baik dengan modul-modul Python lainnya, seperti Pandas dan NumPy.

## 2.7 Plotly

Plotly adalah library untuk pembuatan plot yang tersedia dalam bahasa pemrograman python dan R. Plotly digunakan untuk membuat visualisasi data yang interaktif dan menarik. dari segi kompatibilitas pada diagram, library ini tidak jauh berbeda dengan matplotlib. plot garis, diagram batang, hingga heatmaps mampu digambarkan dengan baik. Salah satu fitur paling kuat Plotly adalah kemampuannya untuk membuat visualisasi data yang dapat diinteraksikan, memungkinkan pengguna untuk menjelajahi data dengan mudah, memfokuskan pada detail, dan memahami hubungan dalam data.

Kelebihan dari library ini antara lain:

- Plotly memungkinkan grafik bisa diperbesar, diperkecil, digeser. Serta mendukung fitur-fitur interaktif seperti tooltip, zoom, dan pan. Ini meningkatkan kemudahan pengguna dan membantu dalam mengungkap pola dan wawasan dalam data.
- Memiliki berbagai macam jenis grafik seperti 3d, histogram, network graph, dan lainnya. Fitur ini bisa menyediakan fleksibilitas untuk memvisualisasikan berbagai jenis data.
- Plotly terintegrasi dengan library python lainnya seperti Pandas, Scipy, Numphy. Memudahkan untuk membuat grafik dari data yang diproses. Grafik disimpan dalam berbagai bentuk format seperti HTML, JSON, SVG, dll. Yang sangat mudah untuk dibagikan.

## 2.8 IO

IO merupakan modul dalam python yang memungkinkan kita mengelola operasi input dan output terkait file. Keuntungan menggunakan modul IO adalah kelas dan fungsi yang tersedia memungkinkan kita memperluas fungsionalitas untuk memungkinkan penulisan ke data Unicode. IO menyediakan kelas-kelas dasar untuk melakukan operasi I/O, baik itu berhubungan dengan file, string, atau objek biner. Ada dua kelas yang berguna dalam modul IO, seperti “BytesIO” dan “StringIO”, yang dapat diakses menggunakan “io.BytesIO” dan “io.StringIO”. “io.StringIO” digunakan untuk membaca data dari string seolah-olah itu adalah

objek file. Ini berguna ketika Anda memiliki data dalam bentuk string dan ingin memprosesnya menggunakan fungsi yang biasanya bekerja dengan berkas (file).

## 2.9 DataFrame

DataFrame adalah struktur data tabular/berbentuk tabel yang umum digunakan dalam analisis data dan pengolahan data di Python, terutama dengan menggunakan library Pandas. DataFrame mirip dengan tabel dalam database atau spreadsheet di mana data disusun dalam baris dan kolom. Setiap kolom dalam DataFrame mewakili satu variabel atau fitur, sementara setiap baris menggambarkan satu entitas atau pengamatan.

DataFrame memudahkan manipulasi dan analisis data karena menyediakan berbagai fungsi dan metode yang memungkinkan kalian untuk melakukan operasi seperti filter, pengurutan, pengelompokan, dan penggabungan data dengan mudah. Dengan menggunakan DataFrame, kalian dapat dengan efisien melakukan transformasi data, menjalankan analisis statistik, dan memvisualisasikan hasilnya.

## 2.10 Int 64

Pada Python, tipe data int64 biasanya merujuk pada bilangan bulat yang memiliki 64-bit. Tipe data int dalam Bahasa python dapat menggambarkan bilangan bulat dengan bobot dan ukuran apa pun, penggunaan tipe data int64 biasanya terhubung dan bekerja dengan library seperti Numpy, yang sering berurusan dengan tipe data yang besifat tetap untuk alasan kinerja dalam komputasi numerik.

## 2.11 Datetime64[ns]

Dalam Bahasa python datetime64[ns] biasanya merujuk kepada tipe data yang menampilkan waktu dan didefinisikan oleh Library Numpy. Tipe data ini digunakan untuk menampilkan tanggal dan waktu dengan sangat presisi nanosekon. Tipe data ini juga digunakan untuk maniplasi suatu deret waktu dan analisi data karena memiliki representasi waktu yang sangat akurat dan efisien. Tetapi ada juga variasi datetime64[m] untuk menit. Dalam Pandas. Presisi dapat digunakan dengan analisis data yang spesifik sesuai kebutuhan pengguna.

## 2.12 Float 64

Float 64 biasanya merujuk terhadapa bilangan 64 bit. Istilah float64 lebih umum digunakan dalam konteks library seperti Numpy. Ini berarti menunjukkan bahwa bilangan desimal dapat memiliki presisi tinggi dan dapat merepresentasikan angka dengan lebih banyak digit dibandingkan dengan tipe data float standar. Biasanya jika menggunakan bilangan

decimal, tipe data float standar sudah cukup. Float 64 digunakan Ketika melakukan operasi numerik yang membutuhkan presisi yang lumayan tinggi untuk penggunaan pengolah sinyal, simulasi numerik, data, etc.

## 2.13 Object

Pada Python, tipe data object adalah tipe data umum yang dapat menampilkan objek yang bersifat sembarang. Tipe data ini memungkinkan kita untuk menyimpan suatu objek python apapun dalam satu variable. Penggunaan tipe data object tidak selalu direkomendasikan untuk dipakai, kecuali jika memang membutuhkan fleksibilitas penuh dalam menyimpan berbagai jenis objek. Tipe data ini memang memberikan fleksibilitas yang lumayan besar namun tipe data object kurang efisien dan lambat karena python harus melakukan pemeriksaan tipe dinamis.

## BAB III

### PEMBAHASAN

#### 3.1 Import Library & Dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt

import plotly.offline as py
import plotly.graph_objects as go
from plotly.offline import init_notebook_mode
init_notebook_mode(connected=True)
from plotly import tools

import warnings
warnings.filterwarnings("ignore", message="numpy.dtype size changed")
```

```
# Proses Import menggunakan url
url =
'https://raw.githubusercontent.com/hananlu/basicPython/master/Dataset/avocadoPrice.csv'
df = pd.read_csv(url)

# Proses Import Lokal (1)
df = pd.read_csv('./avocadoPrice.csv')
df.head()
```

#### Output:

Jnnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany
2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015	Albany
3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015	Albany
4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015	Albany

```
# Proses Import Lokal (2)
from google.colab import files
filenya = files.upload()
```

### Output:

Choose Files avocadoPrice.csv  
 • **avocadoPrice.csv**(text/csv) - 1989197 bytes, last modified: 12/19/2023 - 100% done  
 Saving avocadoPrice.csv to avocadoPrice.csv

```
filenya
import io
df = pd.read_csv(io.StringIO(filenya['avocadoPrice.csv']).decode('utf-
8'))
print(df)
```

### Output:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	\
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	
...	...	...	...	...	...	...	
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	
	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	\
0	48.16	8696.87	8603.62	93.25	0.0	conventional	
1	58.33	9505.56	9408.07	97.49	0.0	conventional	
2	130.50	8145.35	8042.21	103.14	0.0	conventional	
3	72.58	5811.16	5677.40	133.76	0.0	conventional	
4	75.78	6183.95	5986.26	197.69	0.0	conventional	
...	...	...	...	...	...	...	
18244	0.00	13498.67	13066.82	431.85	0.0	organic	
18245	0.00	9264.84	8940.04	324.80	0.0	organic	
18246	727.94	9394.11	9351.80	42.31	0.0	organic	
18247	727.01	10969.54	10919.54	50.00	0.0	organic	
18248	224.53	12014.15	11988.14	26.01	0.0	organic	
	year	region					
0	2015	Albany					
1	2015	Albany					
2	2015	Albany					
3	2015	Albany					
4	2015	Albany					
...	...	...					
18244	2018	WestTexNewMexico					
18245	2018	WestTexNewMexico					
18246	2018	WestTexNewMexico					
18247	2018	WestTexNewMexico					
18248	2018	WestTexNewMexico					

[18249 rows x 14 columns]

## 3.2 Overview Awal Data

```
df.head()
```

Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015 Albany
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015 Albany
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015 Albany
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015 Albany
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015 Albany

Fungsi df.head(), adalah untuk mengembalikan tujuh baris pertama dari DataFrame. Ini dapat digunakan untuk mendapatkan gambaran umum tentang data dalam DataFrame. Fungsi df.head() dapat digunakan untuk tujuan berikut: Untuk memeriksa data dalam DataFrame sebelum melakukan analisis lebih lanjut, Untuk memastikan bahwa DataFrame berisi data yang benar, Untuk menemukan data yang hilang atau tidak valid.

```
df.sample(3)
```

Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
2198	14	2015-09-20	1.56	578500.76	94105.15	380999.77	33739.89	69655.95	68664.57	946.67	44.71	conventional	2015 SanFrancisco
602	30	2015-05-31	0.68	1377670.81	941838.33	315653.39	7287.06	112892.03	99456.40	13421.47	14.16	conventional	2015 DallasFitWorth
4970	30	2016-05-29	0.92	576543.05	183028.56	168419.71	32042.58	193052.20	160356.37	31715.55	980.28	conventional	2016 SanDiego

Fungsi df.sample(), adalah untuk mengembalikan sampel acak dari DataFrame. Sampel ini dapat digunakan untuk mendapatkan gambaran umum tentang data dalam DataFrame, atau untuk melakukan analisis awal. Fungsi df.sample() dapat digunakan untuk tujuan berikut: Untuk memeriksa data dalam DataFrame sebelum melakukan analisis lebih lanjut, Untuk memastikan bahwa DataFrame berisi data yang benar, Untuk menemukan data yang hilang atau tidak valid.

```
# Menampilkan bentuk dari data: (Baris, Kolom)
df.shape
print(f"Jumlah Baris {df.shape[0]}")
print(f"Jumlah Kolumn {df.shape[1]}")
```

```
(18249, 14)
```

```
Jumlah Baris 18249
Jumlah Kolumn 14
```

Fungsi df.shape, adalah untuk mengembalikan ukuran dari DataFrame. Ukuran ini terdiri dari dua nilai jumlah baris dan jumlah kolom. Fungsi df.shape dapat digunakan untuk tujuan berikut untuk memeriksa ukuran DataFrame, Untuk membuat asumsi tentang data dalam DataFrame, untuk membandingkan ukuran DataFrame yang berbeda.

```
df.describe
```

```
<bound method NDFrame.describe of
   Unnamed: 0      Date  AveragePrice  Total Volume    4046    4225 \
0      0  2015-12-27    1.33    64236.62   1036.74   54454.85
1      1  2015-12-20    1.35    54876.98   674.28   44638.81
2      2  2015-12-13    0.93   118220.22   794.70  109149.67
3      3  2015-12-06    1.08    78992.15   1132.00   71976.41
4      4  2015-11-29    1.28    51039.60   941.48   43838.39
...
...     ...     ...
18244    7  2018-02-04    1.63    17074.83   2046.96   1529.20
18245    8  2018-01-28    1.71    13888.04   1191.70   3431.50
18246    9  2018-01-21    1.87    13766.76   1191.92   2452.79
18247   10  2018-01-14    1.93    16205.22   1527.63   2981.04
18248   11  2018-01-07    1.62    17489.58   2894.77   2356.13
   4770  Total Bags  Small Bags  Large Bags  XLarge Bags      type \
0    48.16    8696.87    8603.62     93.25        0.0  conventional
1    58.33    9505.56    9408.07     97.49        0.0  conventional
2   130.50    8145.35    8042.21     103.14        0.0  conventional
3    72.58    5811.16    5677.48     133.76        0.0  conventional
4    75.78    6183.95    5986.26     197.69        0.0  conventional
...
...     ...     ...
18244   0.00    13498.67   13066.82     431.85        0.0  organic
18245   0.00    9264.84    8940.04     324.80        0.0  organic
18246  272.94    9394.11    9351.80     42.31        0.0  organic
18247  272.01   10969.54   10919.54     58.00        0.0  organic
18248  224.53   12014.15   11988.14     26.01        0.0  organic
   year      region
0  2015    Albany
1  2015    Albany
2  2015    Albany
3  2015    Albany
4  2015    Albany
...
...     ...
18244  2018  WestTexNewMexico
18245  2018  WestTexNewMexico
18246  2018  WestTexNewMexico
18247  2018  WestTexNewMexico
18248  2018  WestTexNewMexico
[18249 rows x 14 columns]
```

Fungsi df.describe(), adalah untuk mengembalikan DataFrame yang berisi ringkasan statistik dari setiap kolom dalam DataFrame. Ringkasan ini mencakup informasi berikut: Jumlah nilai non-null, Rata-rata, Standar deviasi, Nilai minimum, Nilai kuartil pertama (Q1), Nilai median, Nilai kuartil ketiga (Q3), Nilai maksimum. Fungsi df.describe() dapat digunakan untuk mendapatkan gambaran umum tentang distribusi data dalam DataFrame. Ini dapat berguna untuk memahami data sebelum melakukan analisis lebih lanjut.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype  
--- 
 0   Unnamed: 0      18249 non-null  int64  
 1   Date            18249 non-null  object 
 2   AveragePrice   18249 non-null  float64 
 3   Total Volume   18249 non-null  float64 
 4   4046            18249 non-null  float64 
 5   4225            18249 non-null  float64 
 6   4770            18249 non-null  float64 
 7   Total Bags     18249 non-null  float64 
 8   Small Bags     18249 non-null  float64 
 9   Large Bags     18249 non-null  float64 
 10  XLarge Bags    18249 non-null  float64 
 11  type            18249 non-null  object 
 12  year            18249 non-null  int64  
 13  region          18249 non-null  object 
dtypes: float64(9), int64(2), object(3)
memory usage: 1.9+ MB
```

Fungsi df.info(), adalah untuk mencetak ringkasan dari sebuah DataFrame Pandas. Ringkasan ini mencakup informasi berikut: Jumlah kolom, Label kolom, Tipe data kolom, Penggunaan memori, Rentang indeks, Jumlah sel di setiap kolom (nilai non-null). Fungsi df.info() dapat digunakan untuk mendapatkan gambaran umum tentang struktur dan konten dari DataFrame. Ini dapat berguna untuk memeriksa data sebelum melakukan analisis lebih lanjut.

```
df.dtypes
```

```
Unnamed: 0      int64
Date           object
AveragePrice   float64
Total Volume   float64
4046          float64
4225          float64
4770          float64
Total Bags    float64
Small Bags    float64
Large Bags    float64
XLarge Bags   float64
type          object
year          int64
region        object
dtype: object
```

Fungsi df.dtypes, adalah untuk mengembalikan DataFrame yang berisi tipe data dari setiap kolom dalam DataFrame. DataFrame ini memiliki dua kolom: column dan dtype. Kolom column berisi nama kolom, dan kolom dtype berisi tipe data kolom. Fungsi df.dtypes dapat digunakan untuk memeriksa tipe data dari setiap kolom dalam DataFrame. Ini dapat berguna untuk memahami struktur dan konten dari DataFrame.

```
df.isnull().sum()
```

```
Unnamed: 0      0
Date           0
AveragePrice   0
Total Volume   0
4046          0
4225          0
4770          0
Total Bags    0
Small Bags    0
Large Bags    0
XLarge Bags   0
type          0
year          0
region        0
dtype: int64
```

Fungsi df.isnull().sum(), adalah untuk menghitung jumlah nilai null dalam setiap kolom dari DataFrame. Nilai null adalah nilai yang tidak terdefinisi atau tidak diketahui. Fungsi df.isnull().sum() dapat digunakan untuk tujuan berikut: Untuk memeriksa apakah ada nilai null dalam DataFrame. Untuk menghitung jumlah nilai null dalam setiap kolom dari DataFrame.

```
df.duplicated().sum()
```

```
0
```

Fungsi `df.duplicated()`, adalah untuk mengembalikan DataFrame Boolean yang menunjukkan apakah setiap baris dalam DataFrame adalah duplikat. Nilai true menunjukkan bahwa baris tersebut merupakan duplikat, dan nilai false menunjukkan bahwa baris tersebut bukan merupakan duplikat. Fungsi `df.duplicated().sum()` adalah untuk menghitung jumlah baris yang merupakan duplikat.

```
df['type'].value_counts()
```

```
conventional    9126
organic        9123
Name: type, dtype: int64
```

Fungsi `df[['__']].value_counts()`, adalah untuk menghitung frekuensi kemunculan setiap nilai unik dalam sebuah kolom DataFrame. Pada contoh diatas menampilkan jumlah frekuensi data pada kolom *type/tipe*, dimana terdapat dua tipe yaitu, *conventional* dan *organic* yang masing-masing berjumlah 9126 dan 9123 buah.

```
df[['region', 'type']].value_counts()
```

```
region      type
Albany     conventional  169
Pittsburgh conventional  169
Roanoke     organic      169
            conventional  169
RichmondNorfolk organic    169
...
HarrisburgScranton conventional  169
GreatLakes     organic      169
            conventional  169
GrandRapids     organic      169
WestTexNewMexico organic    166
Length: 108, dtype: int64
```

Fungsi `df[['__', '__']].value_counts()`, adalah untuk menghitung frekuensi kemunculan setiap nilai unik dalam dua kolom DataFrame. Pada contoh diatas menampilkan jumlah frekuensi data pada kolom *type/tipe* dan *region/wilayah*, dimana terdapat dua tipe yaitu, *conventional* dan *organic* yang tersebar rata ke semua wilayah dengan jumlah 169 buah.

```
df['region'].count()
```

```
18249
```

Fungsi `df['__'].count()`, adalah untuk menghitung jumlah elemen yang tidak bernilai *null* (*non-null* atau *non-missing*) dalam suatu kolom DataFrame. Pada contoh diatas menampilkan jumlah elemen dalam kolom *region/wilayah* penyebaran alpukat sejumlah 18.249 titik daerah.

### 3.3 Visualisasi Data

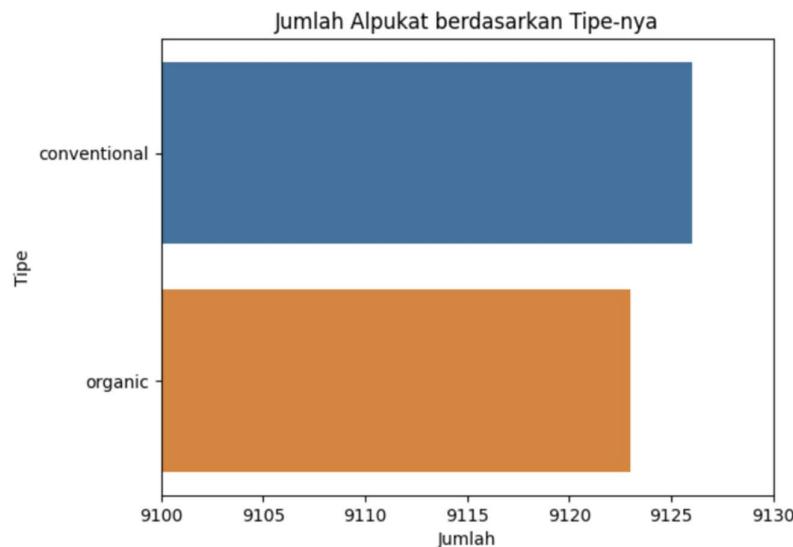
```
sns.countplot(y='type', data=df)

plt.title("Jumlah Alpukat berdasarkan Tipe-nya")
plt.ylabel("Tipe")
plt.xlabel("Jumlah")
plt.xlim(9100, 9130)

plt.show()
```

Kode ini menggunakan seaborn dan matplotlib untuk membuat countplot yang menunjukkan distribusi frekuensi jumlah alpukat berdasarkan jenis ('type') dari DataFrame df. Orientasi horizontal digunakan, di mana sumbu y mewakili jenis alpukat dan panjang batang bar mencerminkan jumlah observasi untuk setiap jenis. Judul plot adalah "Jumlah Alpukat berdasarkan Tipe-nya", dengan label sumbu y "Tipe" dan label sumbu x "Jumlah". Batas sumbu x ditetapkan pada rentang 9100 hingga 9130 untuk fokus pada sebagian data tertentu.

#### Output:



```

sns.countplot(y='year', data=df)

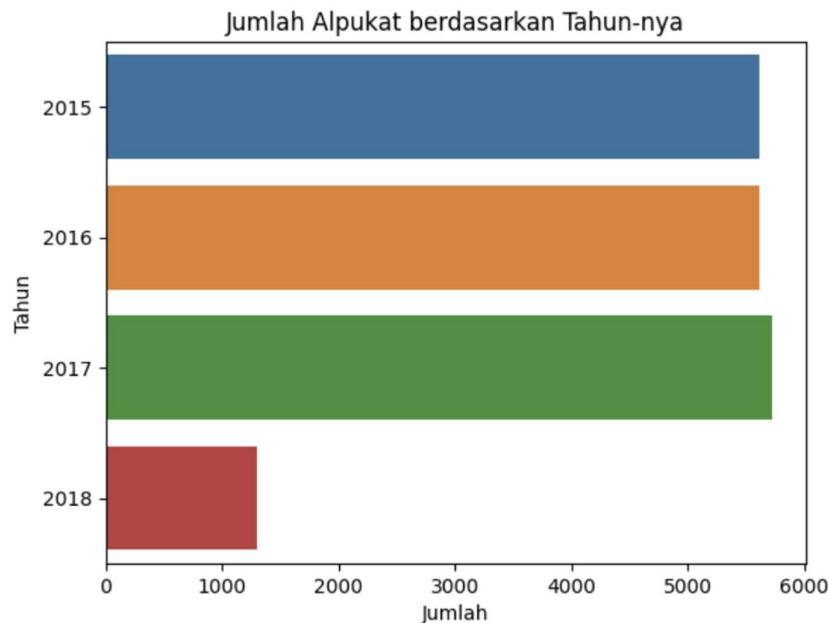
plt.title("Jumlah Alpukat Tiap Tahun-nya")
plt.title("Jumlah Alpukat berdasarkan Tahun-nya")
plt.ylabel("Tahun")
plt.xlabel("Jumlah")

plt.show()

```

Kode ini menggunakan seaborn dan matplotlib untuk membuat countplot yang menunjukkan distribusi frekuensi jumlah alpukat berdasarkan kolom 'year' dari DataFrame df. Sumbu y mewakili tahun dan panjang batang bar mencerminkan jumlah observasi untuk setiap tahun. Judul plot adalah "Jumlah Alpukat Tiap Tahun", dengan label sumbu y "Tahun" dan label sumbu x "Jumlah". Plot ini memberikan gambaran visual tentang sebaran alpukat selama beberapa tahun.

#### **Output:**



```

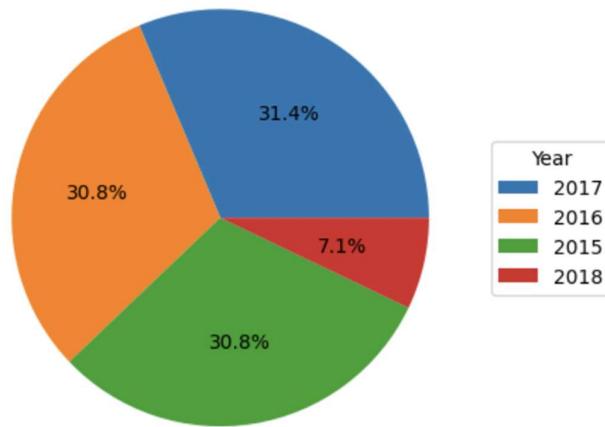
plt.title("Persentase Jumlah Alpukat berdasarkan Tahun-nya")
values = df['year'].value_counts().values
labels = df['year'].value_counts().index
plt.pie(df['year'].value_counts().values, autopct='%.1f%%')
plt.legend(labels, title='Year', loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()

```

Kode ini digunakan untuk membuat diagram pie yang menampilkan persentase jumlah alpukat berdasarkan tahunnya beserta keterangan. Baris pertama untuk menambahkan judul pada plot. Baris kedua dan ketiga untuk mendapatkan nilai dan label yang akan digunakan dalam diagram pie. df['year'].value\_counts().values memberikan nilai yang akan diplot, dan autopct="%.1f%%" menambahkan persentase pada setiap bagian diagram pie. plt.legend() digunakan untuk menambahkan keterangan pada diagram pie. labels adalah label dari setiap bagian, title='Year' memberikan judul untuk keterangan, loc='center left' menentukan lokasi keterangan (di sebelah kiri tengah), dan bbox\_to\_anchor=(1, 0.5) menyesuaikan posisi keterangan.

#### Output:

Persentase Jumlah Alpukat berdasarkan Tahun-nya



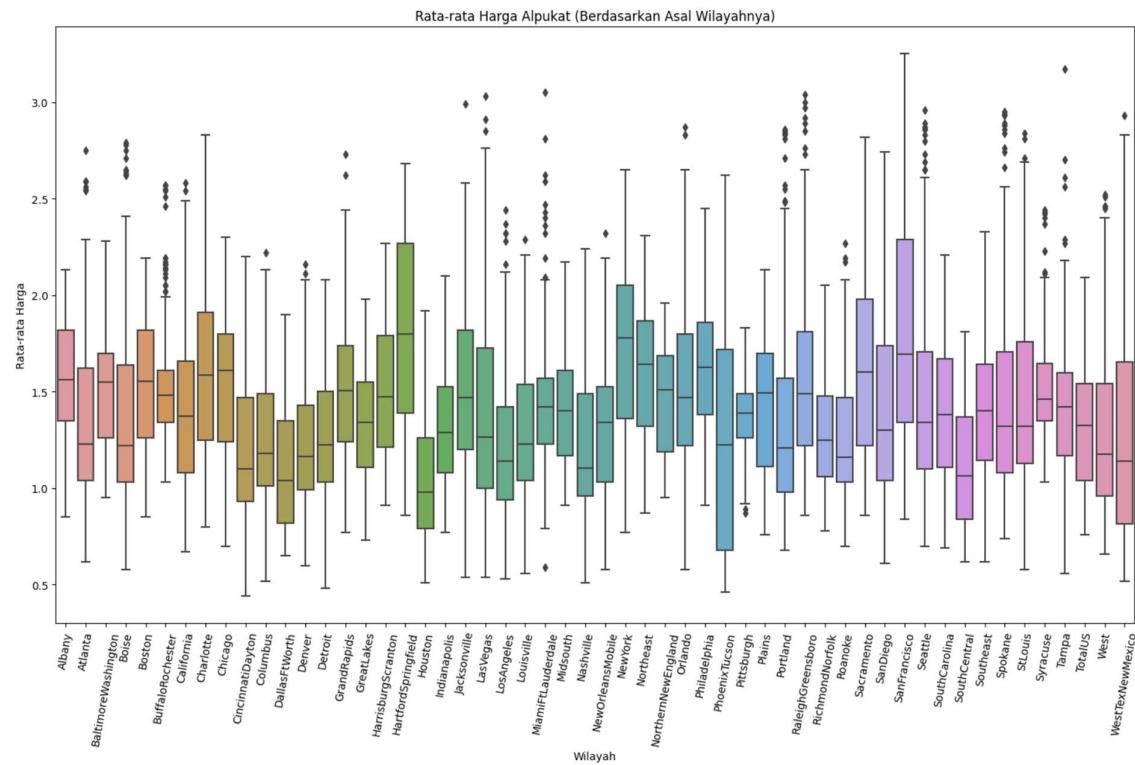
```

plt.figure(figsize=(18,10))
sns.boxplot(x=df['region'], y=df['AveragePrice']);
plt.title("Rata-rata Harga Alpukat (Berdasarkan Asal Wilayahnya)");
plt.ylabel("Rata-rata Harga")
plt.xlabel("Wilayah")
plt.xticks(rotation=80)

```

Kode ini menggunakan seaborn dan matplotlib untuk membuat boxplot yang memvisualisasikan distribusi rata-rata harga alpukat berdasarkan region/wilayah. Baris pertama untuk membuat sebuah figur (figure) dengan ukuran 18x10 inci. Baris kedua untuk menginisiasi Seaborn, membuat boxplot, dengan ‘region’ sebagai variable di sumbu x dan ‘AveragePrice’ sebagai variable di sumbu y.

### Output:



```

plt.figure(figsize=(18, 10))
sns.lineplot(x='year', y="AveragePrice", hue='type', data=df)

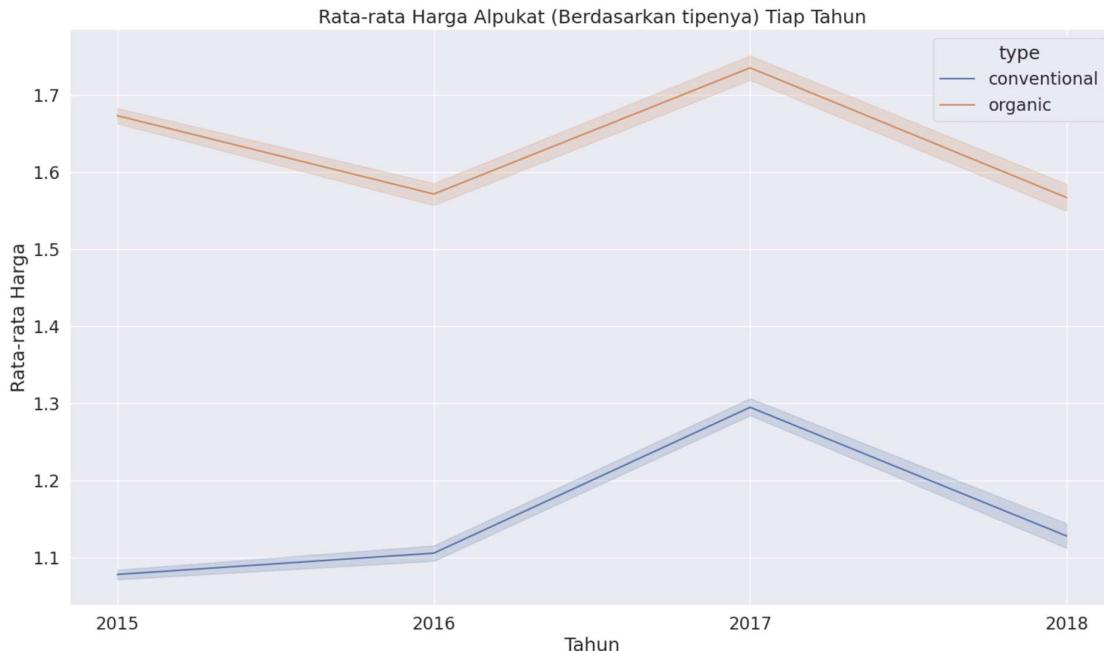
plt.xticks(df['Year'].unique().astype(int))
plt.title("Rata-rata Harga Alpukat (Berdasarkan tipenya) Tiap Tahun")
plt.ylabel("Rata-rata Harga")
plt.xlabel("Tahun")

plt.show()

```

Kode ini digunakan untuk membuat plot/diagram garis menggunakan library Seaborn dan Matplotlib. plt.figure(figsize=(18, 10)) untuk membuat sebuah gambar dengan ukuran 18 unit lebar dan 10 unit tinggi. Pada baris selanjutnya merupakan perintah untuk menggunakan Seaborn dan membuat plot garis. Plot ini memiliki sumbu x yang diidentifikasi oleh kolom year, sumbu y oleh kolom AveragePrice, dan warna garis yang diidentifikasi oleh kolom type. Ini memberikan visualisasi tren harga rata-rata dengan pemisahan berdasarkan tipe. Baris selanjutnya digunakan untuk mengatur posisi label sumbu x dan mengonversinya menjadi integer/bilangan bulat.

### Output:



```

plt.figure(figsize=(18, 10))

sns.lineplot(x="Date", y="AveragePrice", hue='year', data=df)

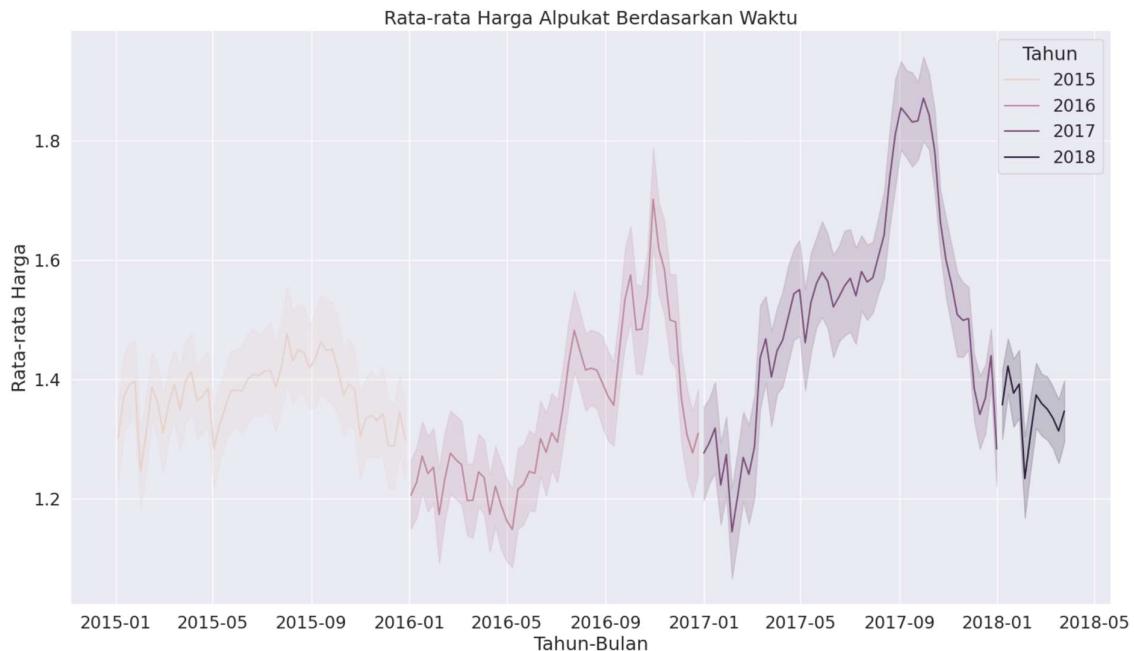
plt.title('Rata-rata Harga Alpukat Berdasarkan Waktu')
plt.xlabel('Tahun-Bulan')
plt.ylabel('Rata-rata Harga')
plt.legend(title='Tahun')

plt.show()

```

Kode ini menggunakan seaborn dan matplotlib untuk membuat line plot yang menampilkan perubahan rata-rata harga alpukat dari waktu ke waktu. Garis-garis berbeda warna, membedakan tahun. Plot ini memberikan gambaran visual tentang tren harga rata-rata alpukat selama beberapa tahun dengan label sumbu x ('Date') untuk menunjukkan waktu dan label sumbu y ('AveragePrice') untuk menunjukkan nilai harga rata-rata. Legenda menandai warna yang mewakili masing-masing tahun.

#### Output:



```
# Konversi tipe data 'Date' jadi datetime
df['Date'] = pd.to_datetime(df['Date'])
df.dtypes
```

Kode ini mengubah kolom 'Date' dalam DataFrame df menjadi tipe data datetime menggunakan pd.to\_datetime(). Tipe data datetime memungkinkan manipulasi dan analisis data berdasarkan tanggal. Setelah perubahan, df.dtypes digunakan untuk menampilkan tipe data setiap kolom dalam DataFrame, memastikan bahwa kolom 'Date' sekarang memiliki tipe data datetime64.

### Output:

```
Unnamed: 0           int64
Date            datetime64[ns]
AveragePrice      float64
Total Volume     float64
4046             float64
4225             float64
4770             float64
Total Bags       float64
Small Bags       float64
Large Bags        float64
XLarge Bags      float64
type             object
year              int64
region            object
Year              int64
Month             int64
Day               int64
dtype: object
```

```
# Membagi nilai didalam kolom Date menjadi: Year, Month, dan Day
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
df.dtypes
```

Kode ini mengekstrak data datetime pada kolom Date di DataFrame dan membaginya menjadi 3 kolom, yaitu: Year, Month dan Day. Setelah menjalankan kode diatas, DataFrame akan memiliki 3 kolom baru yang berisi informasi terkait tahun, bulan dan hari yang didapat dari kolom Date. Operasi df.dtypes digunakan untuk menampilkan tipe data dari setiap kolom di dalam DataFrame setelah penambahan kolom-kolom tersebut.

### Output:

```
Unnamed: 0           int64
Date            datetime64[ns]
AveragePrice      float64
Total Volume     float64
4046             float64
4225             float64
4770             float64
Total Bags       float64
Small Bags       float64
Large Bags        float64
XLarge Bags      float64
type             object
year              int64
region            object
Year              int64
Month             int64
Day               int64
dtype: object
```

```
# Menghapus kolom year
df.drop('year', axis=1, inplace=True)
df.dtypes
```

Kode ini menggunakan metode drop pada DataFrame untuk menghapus kolom dengan label year. Pasalnya, pada DataFrame terdapat dua kolom yaitu year dan Year, Dimana year merupakan data yang diperoleh dari DataFrame awal, dan year diperoleh mengekstrak data Date. Parameter year menunjukkan label kolom yang akan dihapus, dan axis=1 menandakan bahwa penghapusan dilakukan pada kolom (bukan baris). Parameter inplace=True digunakan untuk mengubah DataFrame df asli, tanpa perlu membuat salinan baru. Jika inplace diatur ke False atau tidak disertakan, metode drop akan mengembalikan DataFrame baru dengan kolom yang dihapus.

### Output:

```
Unnamed: 0      int64
Date           datetime64[ns]
AveragePrice   float64
Total Volume   float64
4046           float64
4225           float64
4770           float64
Total Bags     float64
Small Bags     float64
Large Bags     float64
XLarge Bags    float64
type           object
region          object
Year            int64
Month           int64
Day             int64
dtype: object
```

```

plt.figure(figsize=(18,10))
sns.lineplot(x="Month", y="AveragePrice", hue='Year', data=df)

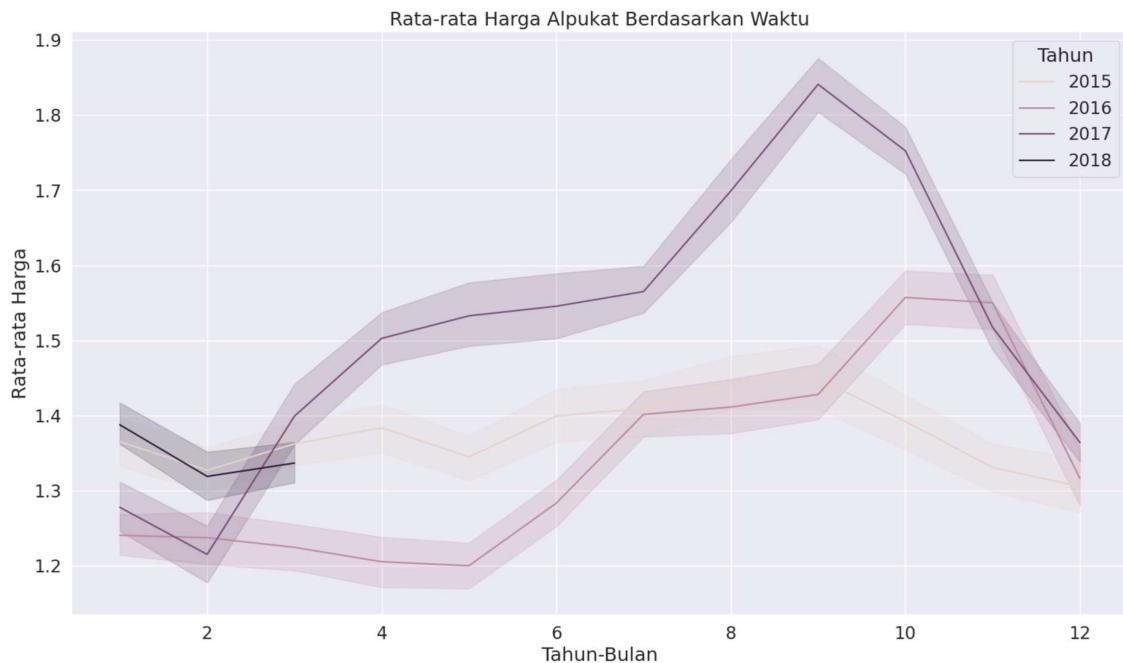
plt.title('Rata-rata Harga Alpukat Berdasarkan Waktu')
plt.xlabel('Tahun-Bulan')
plt.ylabel('Rata-rata Harga')
plt.legend(title='Tahun')

plt.show()

```

Kode ini menggunakan seaborn dan matplotlib untuk membuat line plot yang menunjukkan perubahan rata-rata harga alpukat per bulan. Garis-garis berbeda warna, membedakan tahun. Plot ini memberikan gambaran visual tentang tren perubahan harga rata-rata alpukat tiap bulan, dengan sumbu x ('Month') menunjukkan bulan dan sumbu y ('AveragePrice') menunjukkan nilai harga rata-rata. Ukuran gambar yang besar membantu memastikan keterbacaan dan detail yang baik.

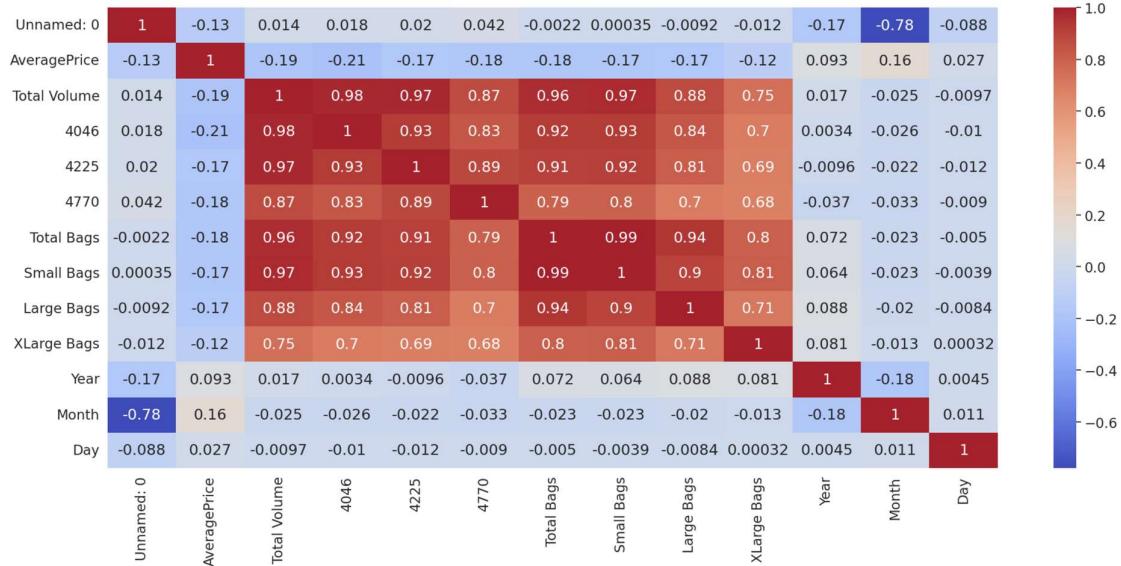
### Output:



```
plt.figure(figsize=(24,10))
sns.heatmap(df.corr(), cmap='coolwarm', annot=True)
```

Kode ini menghasilkan sebuah heatmap yang menunjukkan korelasi antara variable-variabel dalam DataFrame. plt.figure(figsize=(24,10)) membuat sebuah gambar (figure) baru dengan ukuran 24 unit lebar dan 10 unit tinggi. Ini mempengaruhi ukuran keseluruhan dari heatmap yang akan dibuat. cmap='coolwarm' untuk mengatur peta warna (colormap) yang digunakan dalam heatmap. Dalam hal ini, colormap yang digunakan adalah 'coolwarm', yang memiliki variasi warna dari dingin (biru) hingga hangat (merah). Terakhir annot=True digunakan untuk menampilkan nilai korelasi dalam sel. Warna yang lebih terang menunjukkan korelasi yang lebih tinggi, sementara warna yang lebih gelap menunjukkan korelasi yang lebih rendah. Label pada sumbu x dan y biasanya mencantumkan nama variabel atau fitur yang sedang diukur korelasinya.

### Output:



Berdasarkan output heatmap diatas, dapat disimpulkan bahwa tidak semua kolom saling berkorelasi, hanya beberapa saja seperti total volume, total bags, small bags, large bags, XLarge bags, year, month, dan day. Beberapa kolom seperti Region/Wilayah dan Type/Tipe tidak berkorelasi dengan yang lain. Oleh karena itu, untuk memprediksi rata-rata harga alpukat diperlukan *Feature Engineering*.

### 3.4 Feature Engineering

*Feature engineering* merujuk pada proses membuat, memodifikasi, atau memilih fitur (*features*) dari data yang dimiliki dengan tujuan untuk meningkatkan performa model prediktif dalam tugas analisis data atau pembelajaran mesin. Fitur dalam konteks ini adalah variabel atau atribut yang digunakan oleh model untuk membuat prediksi atau menjelaskan pola dalam data. Dengan memahami karakteristik data dan masalah yang dihadapi, praktisi data science dapat menghasilkan fitur-fitur yang lebih informatif untuk mendukung keberhasilan model dalam membuat prediksi yang akurat.

```
# Melakukan Feature Engineering pada categorical Features: "region" dan
# "types"
df['region'].nunique()
```

**Output:**

54

```
df['type'].nunique()
```

**Output:**

2

Dua buah kode diatas digunakan untuk menghitung jumlah nilai unik (*distinct*) dalam kolom 'region' dan 'type' pada DataFrame. df['\_\_']: Ini mengakses kolom dalam DataFrame. .nunique() adalah metode pandas yang menghitung jumlah nilai unik dalam suatu kolom. Berdasarkan output diperoleh 54 region/wilayah dan 2 type/tipe.

```
# Melakukan pengecekan DataFrame
df.dtypes
```

**Output:**

```
Unnamed: 0           int64
Date              datetime64[ns]
AveragePrice      float64
Total Volume      float64
4046              float64
4225              float64
4770              float64
Total Bags        float64
Small Bags        float64
Large Bags        float64
XLarge Bags       float64
type              object
region             object
Year               int64
Month              int64
Day                int64
dtype: object
```

```
# Menghapus kolom 'region' dan 'tanggal'
df_final=pd.get_dummies(df.drop(['region','Date'],axis=1),drop_first=True)
df_final.head()
```

**Output:**

	Unnamed: 0	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	Year	Month	Day	type_organic
0	0	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	2015	12	27	0
1	1	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	2015	12	20	0
2	2	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	2015	12	13	0
3	3	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	2015	12	6	0
4	4	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	2015	11	29	0

```
df_final.tail()
```

**Output:**

	Unnamed: 0	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	Year	Month	Day	type_organic
18244	7	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0	2018	2	4	1
18245	8	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	2018	1	28	1
18246	9	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	2018	1	21	1
18247	10	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	2018	1	14	1
18248	11	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	2018	1	7	1

Kode df\_final.head() dan df\_final.tail() digunakan untuk menampilkan beberapa baris awal (head) dan beberapa baris terakhir (tail) dari DataFrame df\_final. Secara umum, perintah-perintah ini digunakan untuk mendapatkan gambaran singkat tentang struktur dan nilai dalam DataFrame tersebut. Hal ini ditujukan untuk memeriksa struktur, format, dan nilai-nilai di dalamnya.

```
# Membagi dataset menjadi dua set
X=df_final.iloc[:,1:14]
y=df_final['AveragePrice']
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

Kode diatas digunakan untuk membagi dataset menjadi dua set: satu untuk pelatihan model (train) dan satu untuk pengujian model (test). Baris pertama menghasilkan variabel X, yang berisi fitur-fitur dari dataset. iloc[:, 1:14] berarti kita mengambil semua baris (:) dan kolom 1 hingga 13 (indeks 1 hingga 13). Ini bermaksud untuk memilih fitur-fitur dari kolom 1 hingga 13 dari DataFrame. Baris selanjutnya menghasilkan variabel y, yang berisi label atau target dari dataset.

Fungsi train\_test\_split digunakan di sini untuk membagi dataset menjadi empat bagian. X\_train dan y\_train adalah fitur dan label untuk data pelatihan, sedangkan X\_test dan y\_test adalah fitur dan label untuk data pengujian.

```
# Mengevaluasi performa model regresi
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, pred))
print('MSE:', metrics.mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

Kode diatas digunakan untuk menghitung 3 regresi umum pada metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). Diawali dengan mengimpor modul metrics dari scikit-learn, kemudian memulai menghitung MAE, MSE, dan RMSE.

- MAE mengukur rata-rata nilai absolut dari selisih antara nilai sebenarnya dan nilai prediksi. Nilai MAE semakin kecil menunjukkan bahwa model memiliki tingkat kesalahan yang rendah.
- MSE mengukur rata-rata nilai kuadrat dari selisih antara nilai sebenarnya dan nilai prediksi. MSE memberikan "bobot" yang lebih besar pada kesalahan besar. Semakin kecil nilai MSE, semakin baik. MSE yang kecil menunjukkan bahwa model cenderung memiliki kesalahan yang lebih kecil dalam memprediksi nilai.
- RMSE adalah akar kuadrat dari MSE dan memiliki interpretasi yang serupa, tetapi nilai RMSE memiliki satuan yang sama dengan variabel respons. RMSE memberikan gambaran tentang seberapa besar kesalahan model dalam unit yang sama dengan variabel respons.

Seiring meningkatnya nilai MAE, MSE, atau RMSE, kualitas prediksi model menurun, dan sebaliknya. Dalam konteks evaluasi model regresi linear, tujuan utama adalah mencari model dengan kesalahan prediksi yang sekecil mungkin untuk dapat menghasilkan prediksi yang akurat dan berguna.

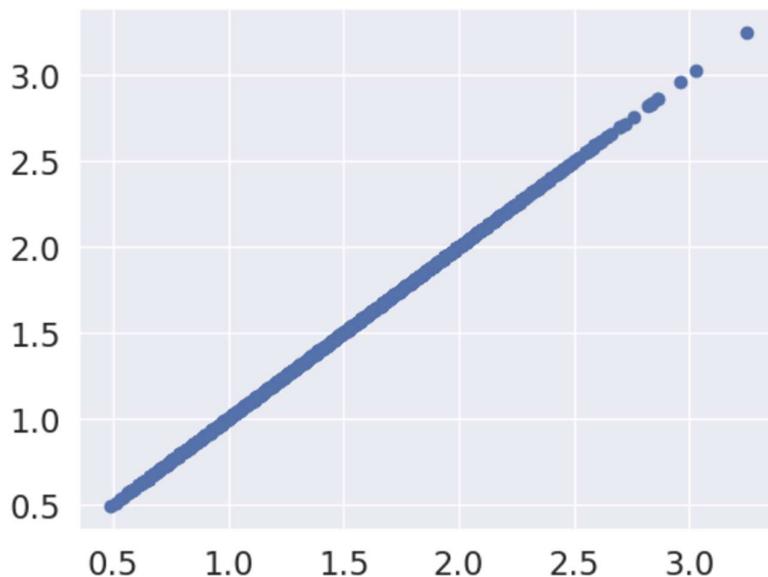
### **Output:**

```
MAE: 3.3334370271697407e-15
MSE: 5.801866300241604e-29
RMSE: 7.616998293449726e-15
```

```
# Membuat scatter plot prediksi  
plt.scatter(x=y_test, y=pred)
```

Kode diatas digunakan untuk membuat scatter plot yang membandingkan nilai sebenarnya ( $y_{test}$ ) dengan nilai prediksi ( $pred$ ). Dengan scatter plot ini, setiap titik mewakili satu pengamatan dalam data pengujian. Jika model melakukan prediksi yang sempurna, semua titik akan terletak pada garis diagonal 45-derajat dari kiri bawah ke kanan atas. Namun, jika terdapat kesalahan prediksi, titik-titik akan tersebar dari garis diagonal.

**Output:**



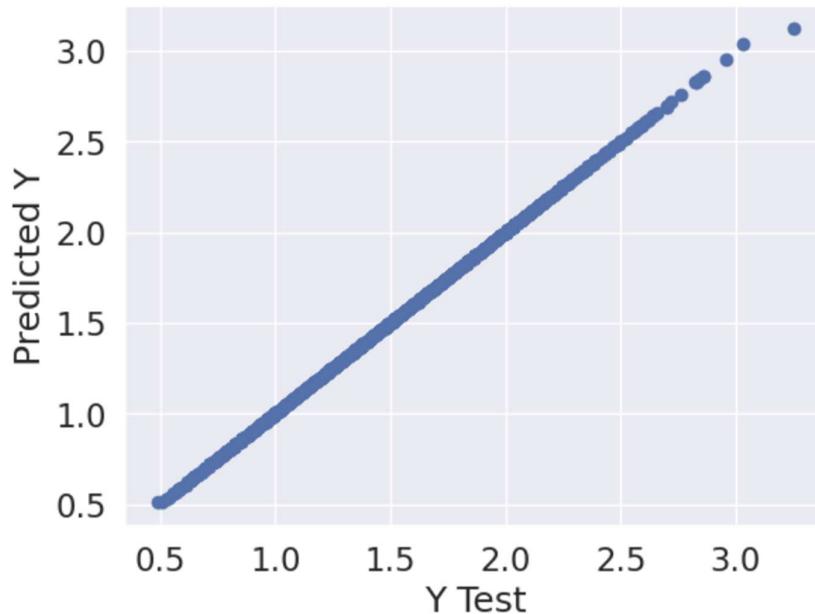
```
# Melatih model regresi berbasis Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor
dtr=DecisionTreeRegressor()
dtr.fit(X_train,y_train)
pred=dtr.predict(X_test)
```

Kode diatas diawali dengan mengimpor DecisionTreeRegressor dari modul tree di scikit-learn. dtr = DecisionTreeRegressor() untuk membuat objek model Regresi Pohon Keputusan. dtr.fit(X\_train, y\_train) untuk melatih model menggunakan data pelatihan (X\_train untuk fitur dan y\_train untuk label). Metode fit menghitung parameter-model berdasarkan data pelatihan. pred = dtr.predict(X\_test) untuk membuat prediksi menggunakan data pengujian (X\_test). Hasil prediksi disimpan dalam variabel pred.

```
# Menampilkan scatter plot prediksi
plt.scatter(x=y_test,y=pred)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

Kode diatas digunakan untuk menampilkan scatter plot yang membandingkan nilai sebenarnya (y\_test) dengan nilai prediksi (pred).

#### Output:



```
# Mengevaluasi performa model regresi
print('MAE:', metrics.mean_absolute_error(y_test, pred))
print('MSE:', metrics.mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

Kode diatas digunakan untuk menampilkan nilai MAE, MSE, dan RMSE untuk mengevaluasi model regresi Decision Tree.

### **Output:**

```
MAE: 5.20547945217651e-05
MSE: 4.849315068493146e-06
RMSE: 0.002202116043375813
```

```
# Melatih model regresi berbasis Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
rdr = RandomForestRegressor()
rdr.fit(X_train,y_train)
pred=rdr.predict(X_test)
```

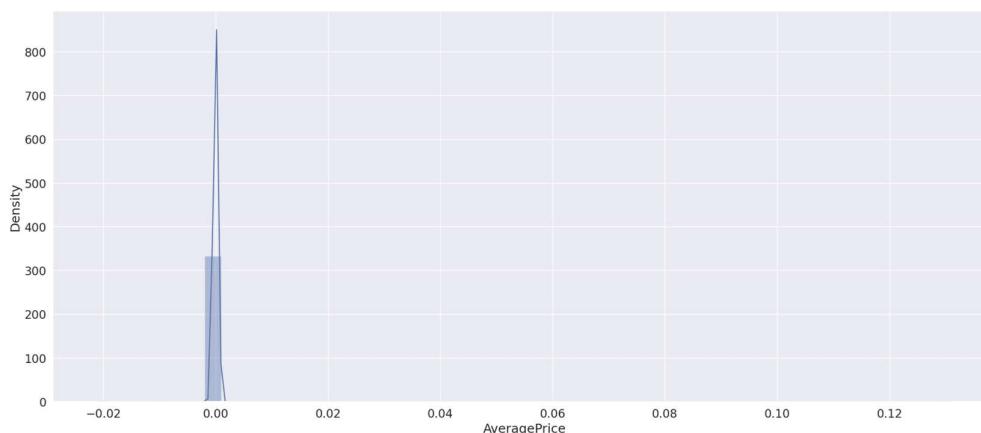
```
# Mengevaluasi performa model regresi
print('MAE:', metrics.mean_absolute_error(y_test, pred))
print('MSE:', metrics.mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

### **Output:**

```
MAE: 7.030136986480106e-05
MSE: 6.59174794520533e-06
RMSE: 0.0025674399594158635
```

```
# Membuat distribution plot dari selisih nilai asli dan nilai prediksi
plt.figure(figsize=(24,10))
sns.distplot((y_test-pred),bins=50)
```

### **Output:**



```
# Membuat perbandingan nilai asli dan nilai prediksi
data = pd.DataFrame({'Y Test':y_test , 'Pred':pred},columns=['Y Test','Pred'])
sns.lmplot(x='Y Test',y='Pred',data=data,palette='rainbow')
data.head()
```

**Output:**

	<b>Y Test</b>	<b>Pred</b>
<b>8604</b>	0.82	0.82
<b>2608</b>	0.97	0.97
<b>14581</b>	1.44	1.44
<b>4254</b>	0.97	0.97
<b>16588</b>	1.45	1.45

