**PRAKTIKUM DATA MINING**



**Dosen Pengampu : Widya Darwin S.Pd,. M.Pd.T**

**Nama : Najwa Alawiyah Siregar**

**NIM :22346040**

# INFORMATIKA

**DEPARTEMEN ELEKTRONIKA, FAKULTAS TEKNIK**

**UNIVERSITAS NEGERI PADANG**

# 2024/2025

**Laporan Hasil Analisis Harga Produk dalam Dataset Penjualan Tembakau**

Berikut kodenya:

import pandas as pd

import matplotlib.pyplot as plt

def main():

    # Coba memuat dataset CSV dengan penanganan kesalahan, serta menggunakan delimiter yang sesuai

    try:

        df = pd.read\_csv('penjualan\_tembakau.csv', delimiter=';')  # Menambahkan delimiter

    except FileNotFoundError:

        print("File 'penjualan\_tembakau.csv' tidak ditemukan.")

        return

    # Tampilkan nama-nama kolom untuk memverifikasi apakah kolom yang diperlukan ada

    print("Nama kolom dalam dataset:")

    print(df.columns)

    # Tampilkan 5 data pertama dan 5 data terakhir

    print("\n5 data pertama:")

    print(df.head())

    print("\n5 data terakhir:")

    print(df.tail())

    # Tampilkan jumlah produk secara keseluruhan

    jumlah\_produk = len(df)

    print(f"\nJumlah produk: {jumlah\_produk}")

    # Verifikasi apakah kolom 'category', 'name', dan 'In Stock' atau 'penjualan' ada dalam dataset sebelum melanjutkan

    if 'category' in df.columns and 'name' in df.columns:

        # Kelompokkan data berdasarkan kategori dan hitung total produk per kategori

        total\_produk\_per\_kategori = df.groupby('category')['name'].count()

        print("\nTotal produk per kategori:")

        print(total\_produk\_per\_kategori)

        # Menghitung total stok atau penjualan per kategori

        if 'In Stock' in df.columns:  # Menghitung total stok per kategori

            total\_stok\_per\_kategori = df.groupby('category')['In Stock'].sum()

            print("\nTotal stok per kategori:")

            print(total\_stok\_per\_kategori)

            # Hapus nilai negatif jika ada

            total\_stok\_per\_kategori = total\_stok\_per\_kategori[total\_stok\_per\_kategori >= 0]

            # Visualisasi: Bar chart total stok per kategori

            plt.figure(figsize=(10, 6))

            total\_stok\_per\_kategori.plot(kind='bar', color='skyblue')

            plt.title('Total Stok per Kategori')

            plt.xlabel('Kategori')

            plt.ylabel('Total Stok')

            plt.xticks(rotation=45)

            plt.show()

            # Visualisasi: Pie chart proporsi stok per kategori

            plt.figure(figsize=(8, 6))

            total\_stok\_per\_kategori.plot(kind='pie', autopct='%1.1f%%', colors=['lightblue', 'lightgreen', 'lightcoral'])

            plt.title('Proporsi Stok per Kategori')

            plt.ylabel('')

            plt.show()

        elif 'penjualan' in df.columns:  # Menghitung total penjualan per kategori

            total\_penjualan\_per\_kategori = df.groupby('category')['penjualan'].sum()

            print("\nTotal penjualan per kategori:")

            print(total\_penjualan\_per\_kategori)

            # Hapus nilai negatif jika ada

            total\_penjualan\_per\_kategori = total\_penjualan\_per\_kategori[total\_penjualan\_per\_kategori >= 0]

            # Visualisasi: Bar chart total penjualan per kategori

            plt.figure(figsize=(10, 6))

            total\_penjualan\_per\_kategori.plot(kind='bar', color='lightgreen')

            plt.title('Total Penjualan per Kategori')

            plt.xlabel('Kategori')

            plt.ylabel('Total Penjualan')

            plt.xticks(rotation=45)

            plt.show()

            # Visualisasi: Pie chart proporsi penjualan per kategori

            plt.figure(figsize=(8, 6))

            total\_penjualan\_per\_kategori.plot(kind='pie', autopct='%1.1f%%', colors=['lightblue', 'lightgreen', 'lightcoral'])

            plt.title('Proporsi Penjualan per Kategori')

            plt.ylabel('')

            plt.show()

    else:

        print("\nKolom 'category' atau 'name' tidak ditemukan dalam dataset.")

        return

    # Tugas 2: Analisis Data dengan Statistik Deskriptif

    if 'penjualan' in df.columns:

        print(f"Rata-rata penjualan: {df['penjualan'].mean()}")

        print(f"Median penjualan: {df['penjualan'].median()}")

        print(f"Modus penjualan: {df['penjualan'].mode()[0]}")

        print(f"Standar deviasi penjualan: {df['penjualan'].std()}")

        # Membuat box plot penjualan

        df['penjualan'].plot(kind='box', figsize=(8, 6))

        plt.title('Distribusi Penjualan')

        plt.xlabel('Penjualan')

        plt.show()

    # Tugas 4: Analisis Harga Produk

    if 'harga' in df.columns:  # Pastikan kolom 'harga' ada

        print(f"\nRata-rata harga: {df['harga'].mean()}")

        print(f"Median harga: {df['harga'].median()}")

        print(f"Modus harga: {df['harga'].mode()[0]}")

        print(f"Standar deviasi harga: {df['harga'].std()}")

        # Membuat box plot harga

        df['harga'].plot(kind='box', figsize=(8, 6))

        plt.title('Distribusi Harga Produk')

        plt.xlabel('Harga')

        plt.show()

    # Tugas 3: Membersihkan Data

    # Cek missing values

    print("Missing values:")

    print(df.isna().sum())

    # Hapus data dengan missing values

    df = df.dropna()

    # Ganti missing values dengan nilai rata-rata (jika diperlukan)

    if 'penjualan' in df.columns:

        df['penjualan'] = df['penjualan'].fillna(df['penjualan'].mean())

    if 'harga' in df.columns:

        df['harga'] = df['harga'].fillna(df['harga'].mean())

    # Cek duplikat data

    print("\nDuplikat data:")

    print(df.duplicated().sum())

    # Hapus duplikat data

    df = df.drop\_duplicates()

# Memanggil fungsi main

if \_\_name\_\_ == "\_\_main\_\_":

    main()

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

def clustering\_data(df):

    # Memilih kolom numerik yang tersedia untuk clustering

    if 'Basic - Price' in df.columns and 'In Stock' in df.columns:

        # Menggunakan hanya kolom numerik yang relevan untuk clustering

        data = df[['Basic - Price', 'In Stock']].fillna(0)  # Mengisi NaN dengan 0

        scaler = StandardScaler()

        data\_scaled = scaler.fit\_transform(data)

        # Menerapkan KMeans

        kmeans = KMeans(n\_clusters=3, random\_state=42)

        df['cluster'] = kmeans.fit\_predict(data\_scaled)

        # Menampilkan hasil clustering

        print("Distribusi data per kluster:")

        print(df['cluster'].value\_counts())

        # Visualisasi

        plt.figure(figsize=(8, 6))

        plt.scatter(data\_scaled[:, 0], data\_scaled[:, 1], c=df['cluster'], cmap='viridis')

        plt.xlabel('Basic - Price (scaled)')

        plt.ylabel('In Stock (scaled)')

        plt.title('Hasil Clustering dengan KMeans')

        plt.show()

    else:

        print("Kolom yang dibutuhkan untuk clustering tidak ada.")

def main():

    # Memuat dataset

    try:

        df = pd.read\_csv('penjualan\_tembakau.csv', delimiter=';')

    except FileNotFoundError:

        print("File 'penjualan\_tembakau.csv' tidak ditemukan.")

        return

    # Pastikan df terdefinisi sebelum dipanggil dalam clustering\_data

    clustering\_data(df)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

def classification\_data(df):

    # Menyiapkan label berdasarkan stok (misalnya, 0 untuk rendah, 1 untuk tinggi)

    df['stock\_label'] = (df['In Stock'] > 5).astype(int)

    # Memilih fitur untuk klasifikasi

    features = ['Basic - Price', 'Cost Amount']

    X = df[features].fillna(0)  # Mengisi NaN dengan 0 jika diperlukan

    y = df['stock\_label']

    # Membagi data menjadi training dan testing

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)

    # Menggunakan Decision Tree untuk klasifikasi

    clf = DecisionTreeClassifier(random\_state=0)

    clf.fit(X\_train, y\_train)

    # Prediksi dan akurasi

    y\_pred = clf.predict(X\_test)

    print(f"Akurasi klasifikasi stok: {accuracy\_score(y\_test, y\_pred) \* 100:.2f}%")

def main():

    # Memuat dataset

    try:

        df = pd.read\_csv('penjualan\_tembakau.csv', delimiter=';')

    except FileNotFoundError:

        print("File 'penjualan\_tembakau.csv' tidak ditemukan.")

        return

    # Memanggil fungsi klasifikasi setelah data dimuat

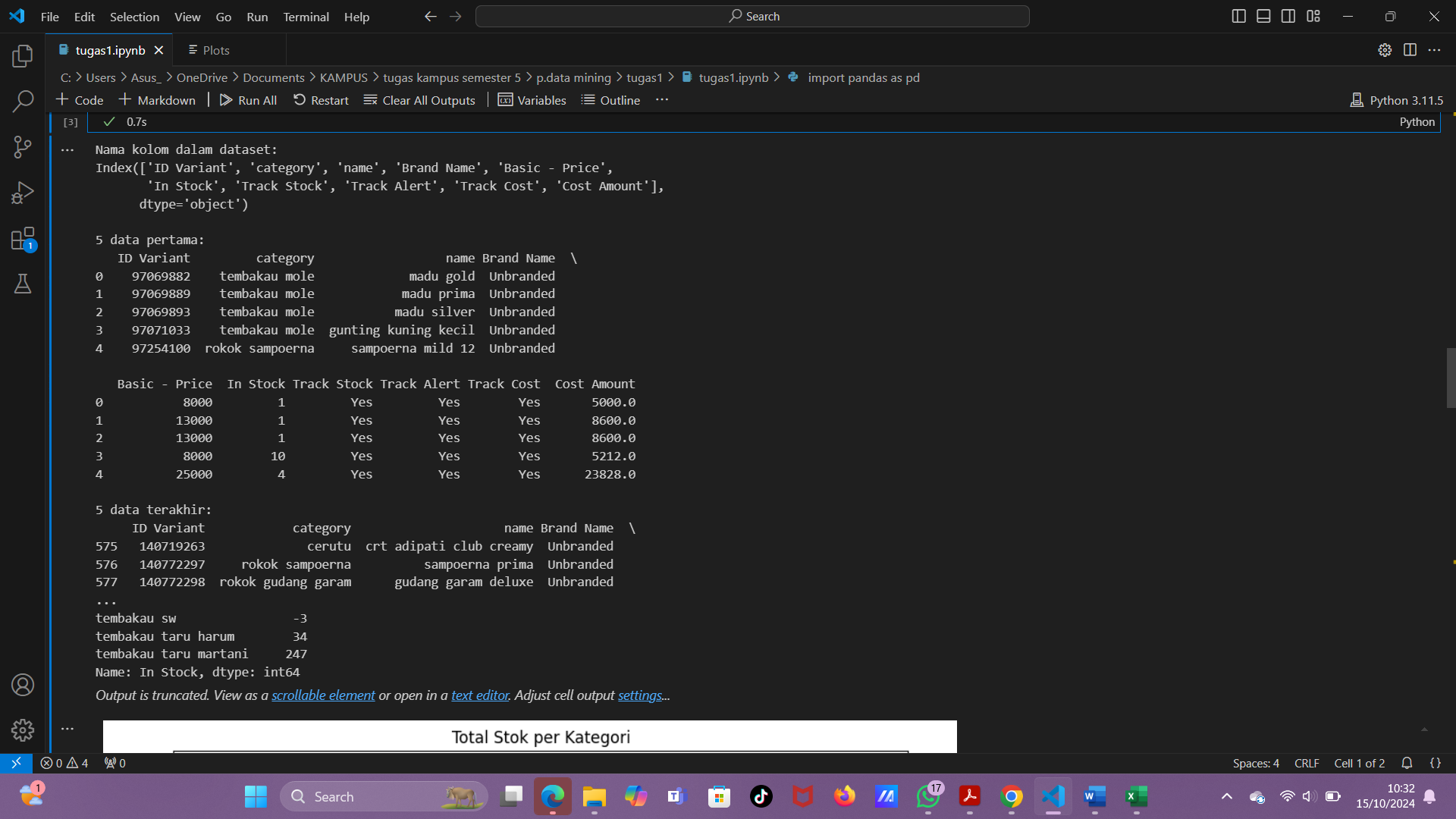
    classification\_data(df)

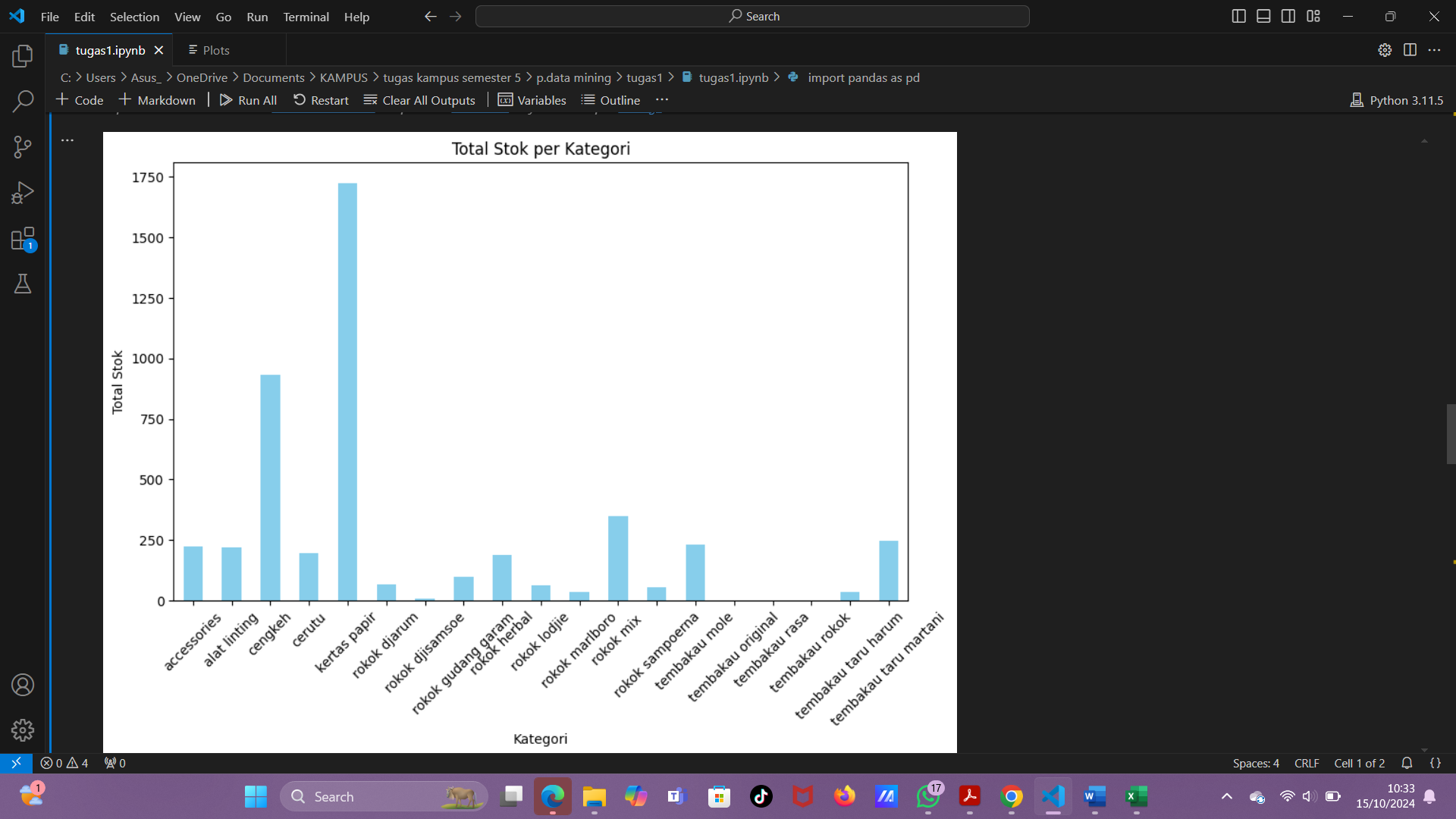
# Memanggil fungsi main

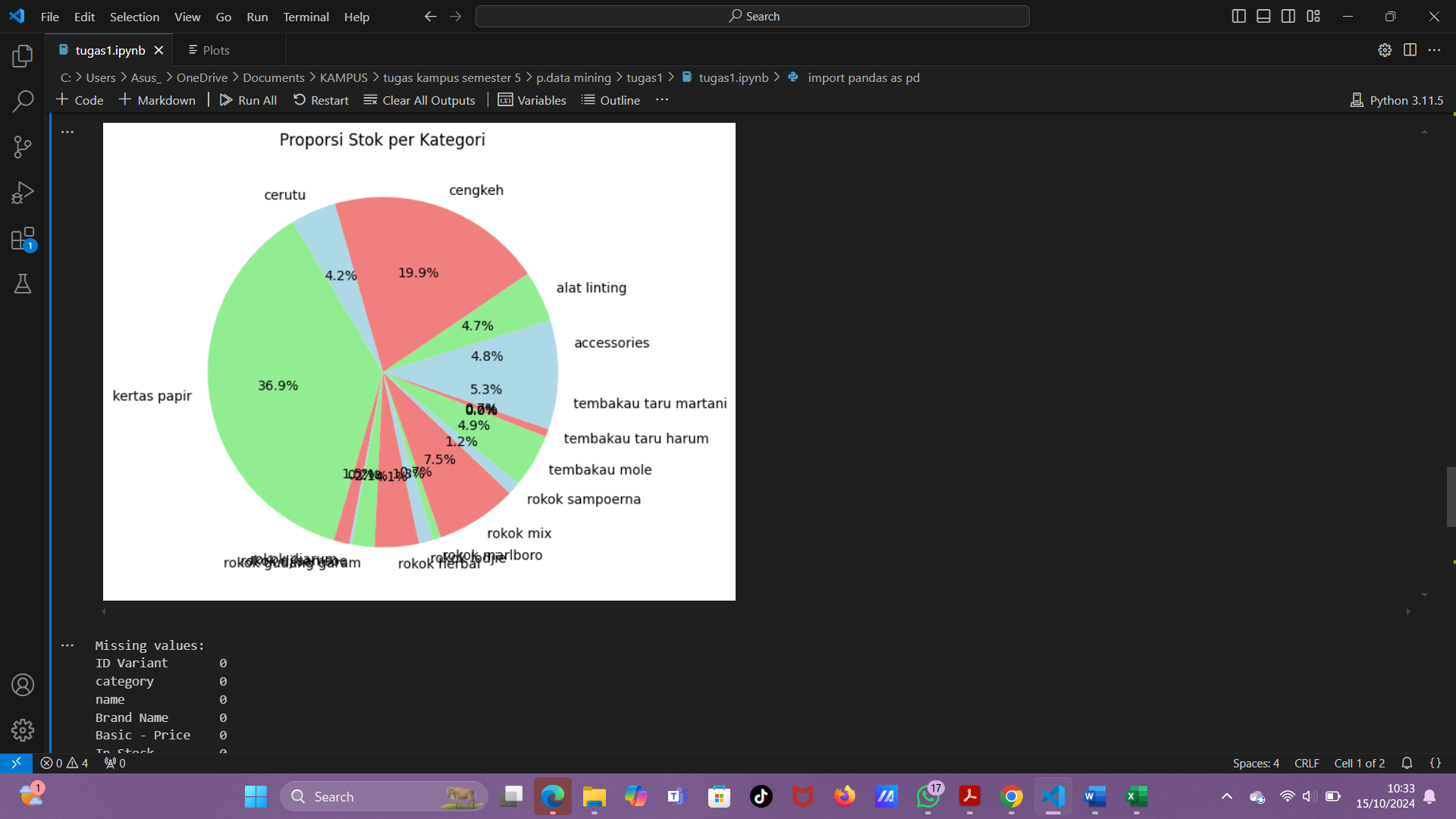
if \_\_name\_\_ == "\_\_main\_\_":

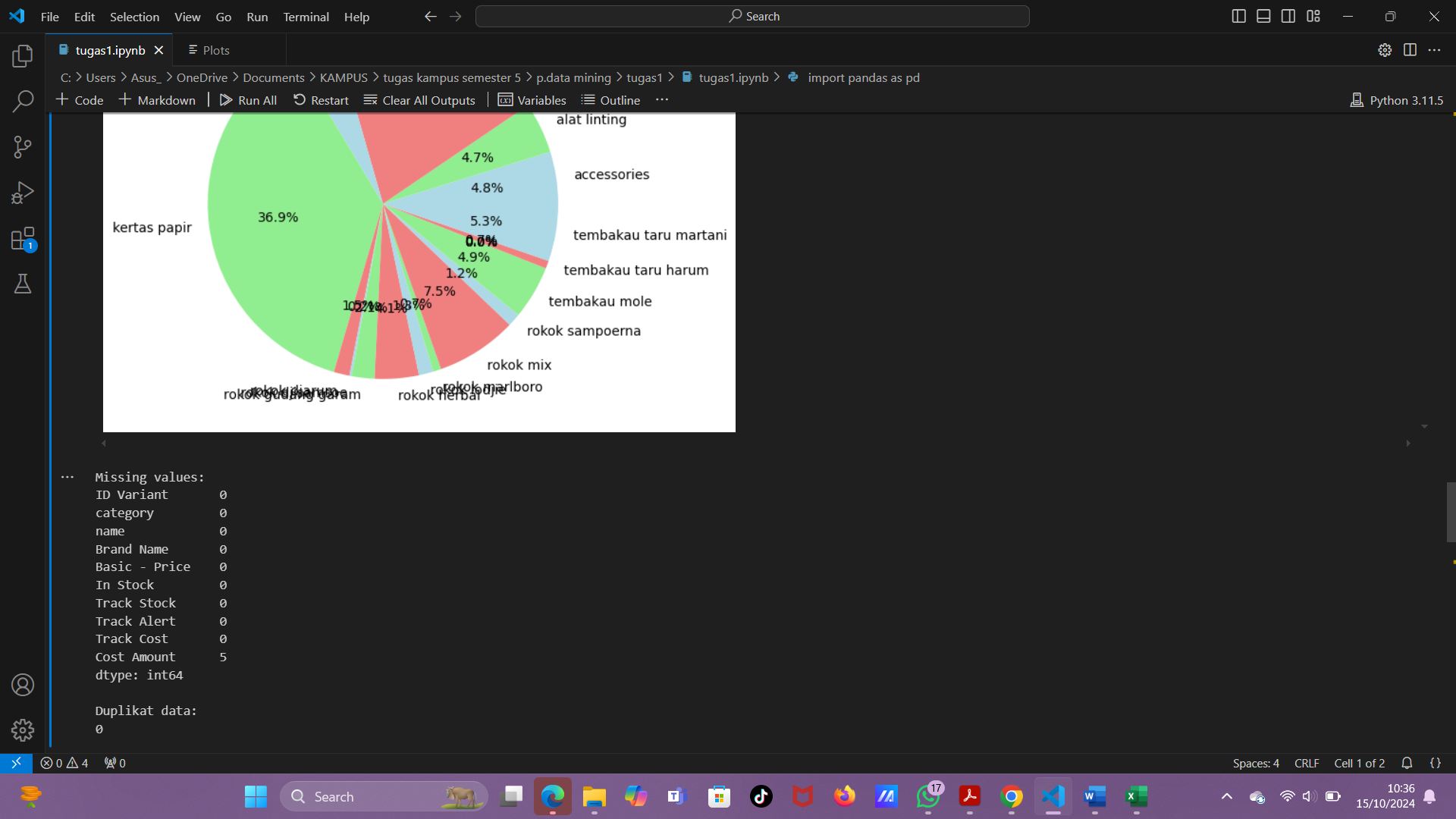
    main()

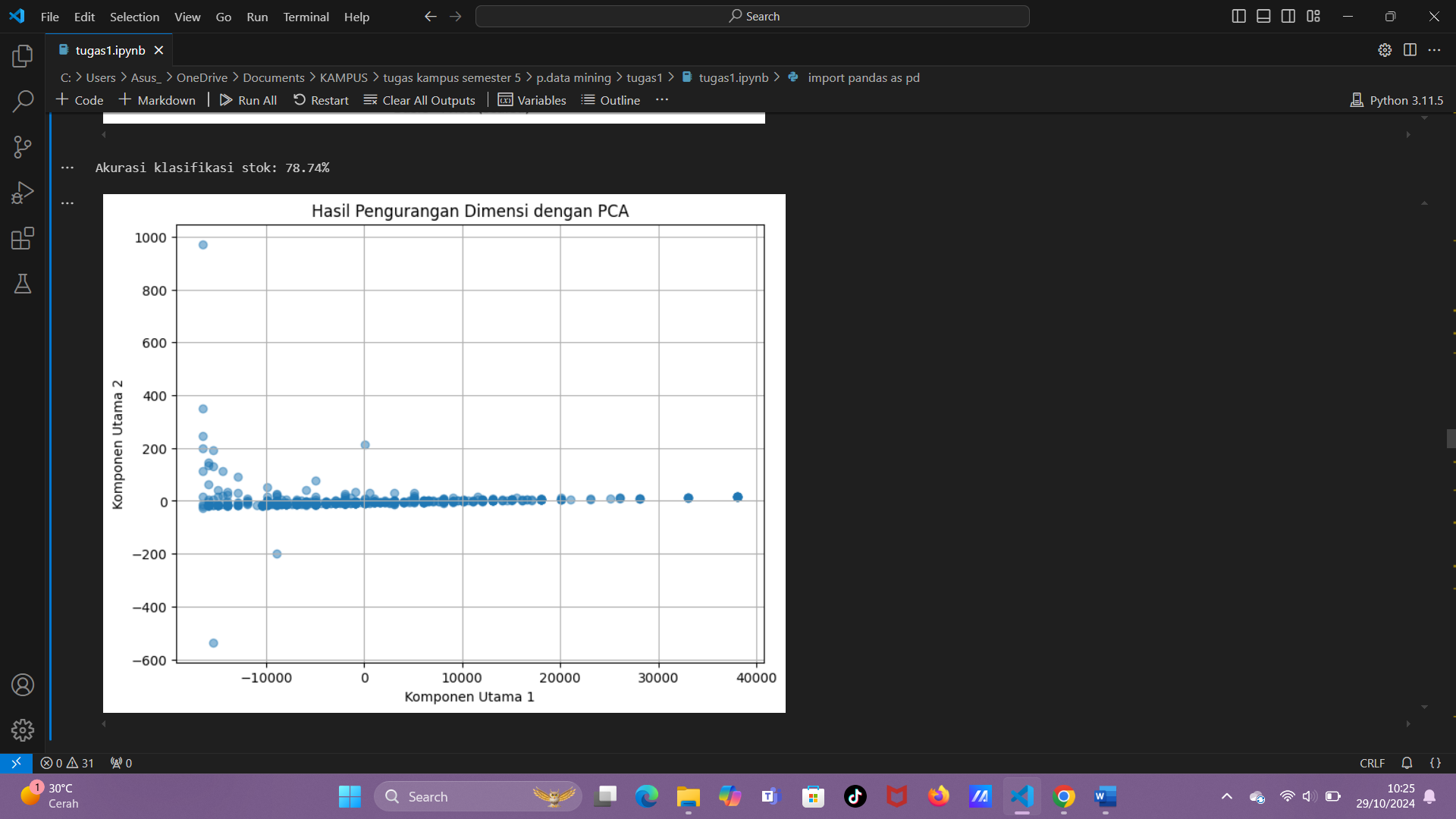
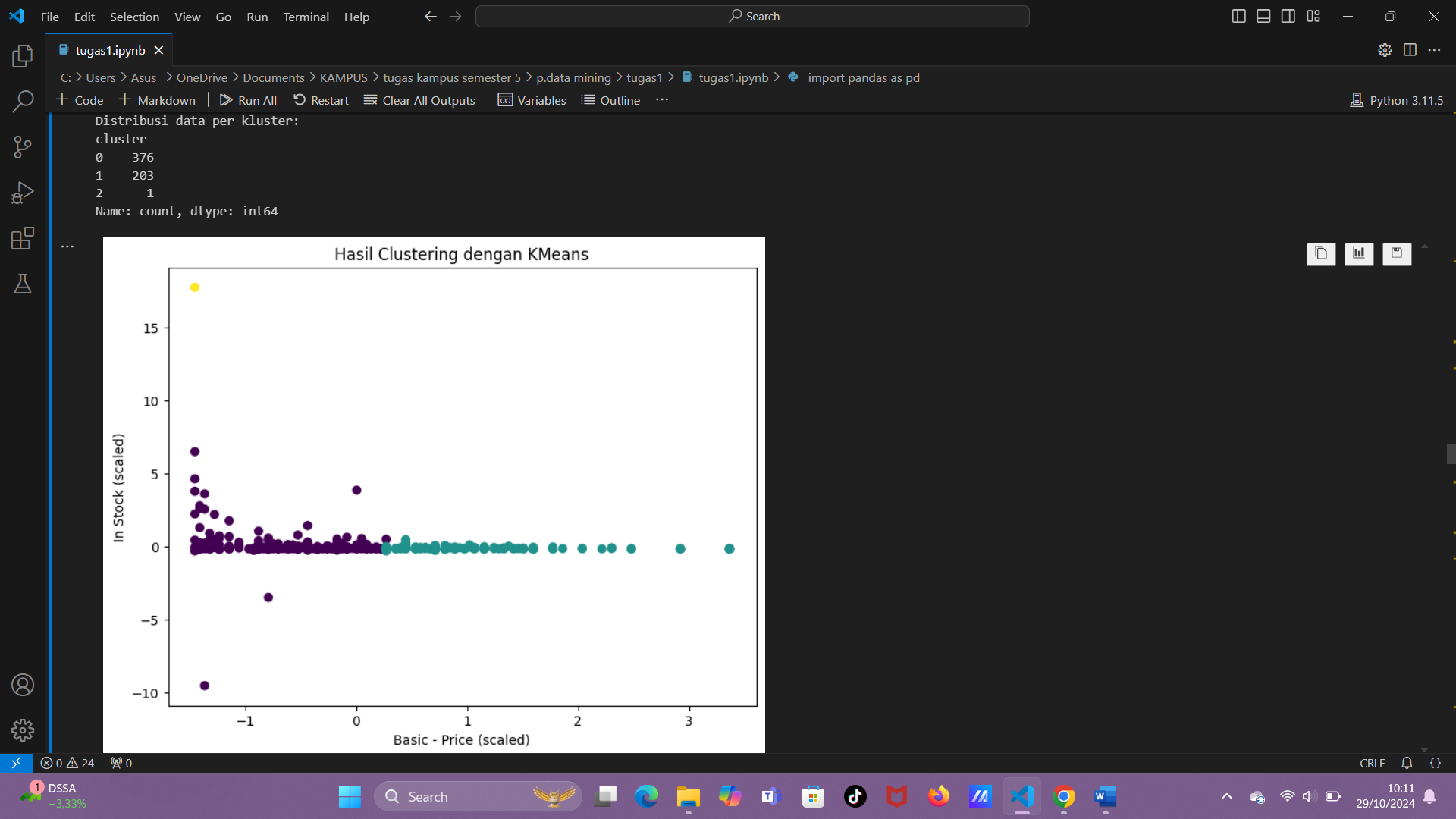
Dan Ini Hasilnya:











Penjelasannya:

Berdasarkan kode di atas, telah terdapat:

1. **Eksplorasi Data**:
   * Menampilkan nama kolom, lima data pertama dan terakhir.
   * Menghitung jumlah produk.
   * Visualisasi total stok dan penjualan per kategori dalam bentuk *bar chart* dan *pie chart*.
   * Analisis statistik deskriptif pada kolom penjualan dan harga (jika ada).
   * Pembersihan data, seperti menghapus *missing values*, mengganti *missing values* dengan nilai rata-rata, dan menghapus duplikasi.
2. **Klasifikasi**:
   * Fungsi classification\_data() yang menggunakan *Decision Tree* untuk memprediksi stock\_label (0 untuk stok rendah dan 1 untuk stok tinggi) berdasarkan fitur Basic - Price dan Cost Amount.
   * Data dibagi menjadi set pelatihan dan pengujian, dan hasil akurasi ditampilkan.
3. **Clustering**:
   * Fungsi clustering\_data() yang melakukan *clustering* menggunakan *KMeans* dengan 3 kluster berdasarkan fitur Basic - Price dan In Stock.
   * Data diskalakan dengan StandardScaler untuk memperbaiki distribusi sebelum clustering.
   * Hasil clustering ditampilkan dalam bentuk distribusi data per kluster dan divisualisasikan dengan *scatter plot*.

Secara keseluruhan, ketiga proses analisis — *eksplorasi data*, *klasifikasi*, dan *clustering* — telah ada dalam dataset.