**DOKUMENTASI PROYEK AKHIR PENAMBANGAN DATA**

**PENERAPAN ALGORITMA APRIORI DALAM PENGEMBANGAN APLIKASI PENAMBANGAN DATA**



Disusun Oleh :

Sie Deta Dirganjaya

NIM A11.2022.13960

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS DIAN NUSWANTORO**

**SEMARANG**

**2023**

# DAFTAR ISI

[DAFTAR ISI ii](#_Toc154765326)

[Dataset 1](#_Toc154765327)

[Permasalahan 1](#_Toc154765328)

[Tujuan 2](#_Toc154765329)

[Model dan Alur 2](#_Toc154765330)

[1) Desain Aplikasi 2](#_Toc154765331)

[2) Library 2](#_Toc154765332)

[3) Input Data 3](#_Toc154765333)

[4) Filterisasi Data 4](#_Toc154765334)

[5) Pembuatan Frequent Itemsets dan Association Rules 7](#_Toc154765335)

[6) Visualisasi Data 9](#_Toc154765336)

[Performa Model / Uji Performa Model 14](#_Toc154765337)

[Proses Deployment 14](#_Toc154765338)

[Kesimpulan dan Rekomendasi 17](#_Toc154765339)

## Dataset

Dataset yang digunakan bersifat PRIVATE bersumber dari perusahaan CV Sanjaya Utama dan sudah mendapatkan izin persetujuan untuk digunakan sebagai dataset dalam proyek akhir penambangan data.

Dataset yang diberikan dalam file sjy.csv terdiri dari beberapa fitur, yaitu "TGL" (tanggal), "BARANG" (nama barang), "QTY" (jumlah), "HARGA (RP)" (harga dalam Rupiah), "VAT (11%)" (nilai PPN 11%), dan "DEBIT" (jumlah debit). Data ini mencerminkan transaksi pembelian barang. Setiap debit dan tanggal mewakili satu transaksi dengan rincian barang, jumlah, dan harga.

Dalam dataset ini, fitur-fitur utamanya adalah:

1. TGL (tanggal): Menunjukkan tanggal transaksi dilakukan.
2. BARANG (nama barang): Merupakan nama barang yang dibeli.
3. QTY (jumlah): Menunjukkan jumlah barang yang dibeli.
4. HARGA (RP) (harga dalam Rupiah): Menunjukkan harga barang dalam mata uang Rupiah.
5. VAT (11%) (nilai PPN 11%): Menunjukkan nilai PPN 11% untuk setiap transaksi.
6. DEBIT (jumlah debit): Merupakan informasi terkait jumlah debit untuk setiap transaksi.

## Permasalahan

Pandemi global yang dikenal dengan Covid-19 memasuki Indonesia pada 3 tahun lalu yaitu tahun 2020. Pandemi Covid-19 telah memberikan dampak yang signifikan terhadap perusahaan dan ekonomi di Indonesia termasuk CV Sanjaya Utama. CV Sanjaya Utama adalah perusahaan yang dikenal dengan kelengkapan alat dan bahan pengemasan di Semarang.

Dampak pandemi Covid-19 terhadap CV Sanjaya Utama:

* Penurunan permintaan karena diberlakukannya lockdown.
* Gangguan rantai pasokan karena kesulitan mendapatkan bahan baku dan komponen.
* Kenaikan biaya tenaga kerja, biaya bahan baku, dan biaya transportasi.

Meskipun pada tahun 2023 ini CV Sanjaya Utama sudah mengalami peningkatan sejak pandemi, namun tetap saja pendapatan perusahaan tidak sebesar dan seramai jaman dahulu sebelum terjadinya pandemi. Tetapi perusahaan memiliki data historis transaksi penjualan dari bulan ke bulan, namun sayangnya hanya digunakan sebagai laporan mingguan dan bulanan saja. Jika semakin lama dibiarkan, maka akan terjadi pertumbuhan data yang menimbulkan kaya data namun miskin informasi. Data yang tidak diolah ini hanya akan mengakibatkan penumpukan data yang tidak bermanfaat. Perusahaan juga ingin meningkatkan pendapatan dengan membuat promo produk namun tidak tahu harus menggunakan pasangan produk apa yang paling cocok untuk dipromosikan.

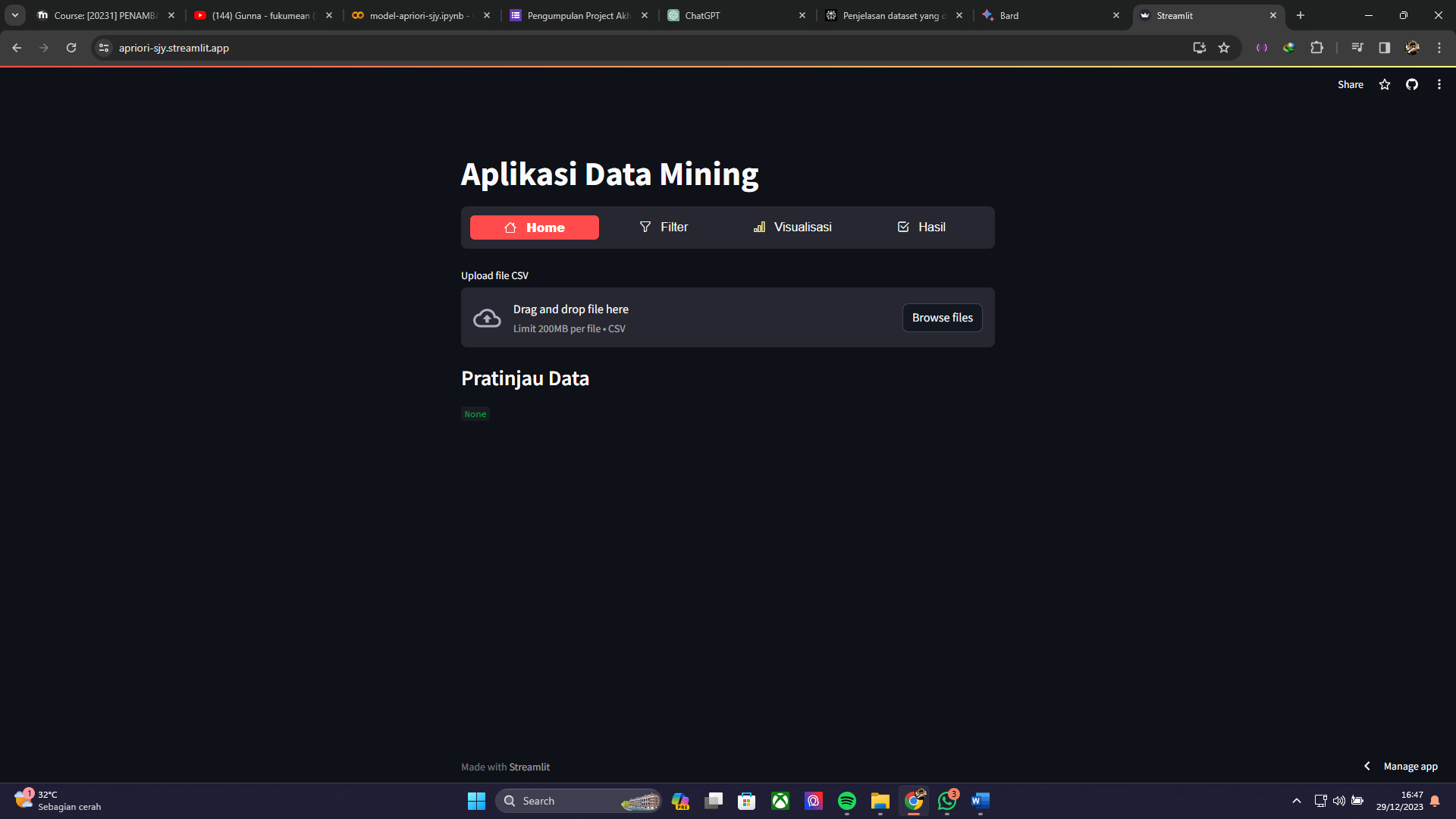
## Tujuan

Dari permasalahan yang sudah dibahas sebelumnya, tujuan saya membangun aplikasi ini adalah untuk mendapatkan pola transaksi yang dilakukan oleh pelanggan, pola tersebut nantinya diolah menjadi rekomendasi promo dari pasangan barang untuk dapat meningkatkan pendapatan bagi perusahaan. Harapannya setelah aplikasi ini diserahkan kepada perusahaan, mereka dapat memperbaiki dan mengingkatkan ekonomi perusahaan melalui promo yang dibuat.

## Model dan Alur

Aplikasi ini menggunakan Algoritma Apriori dan dibangun menggunakan bahasa pemograman Python dengan framework Streamlit. Berikut adalah model dan alur aplikasi:

### Desain Aplikasi



Aplikasi memiliki 4 bagian yaitu Home, Filter, Visualisasi, Hasil.

### Library

# Import library yang diperlukan

import pandas as pd

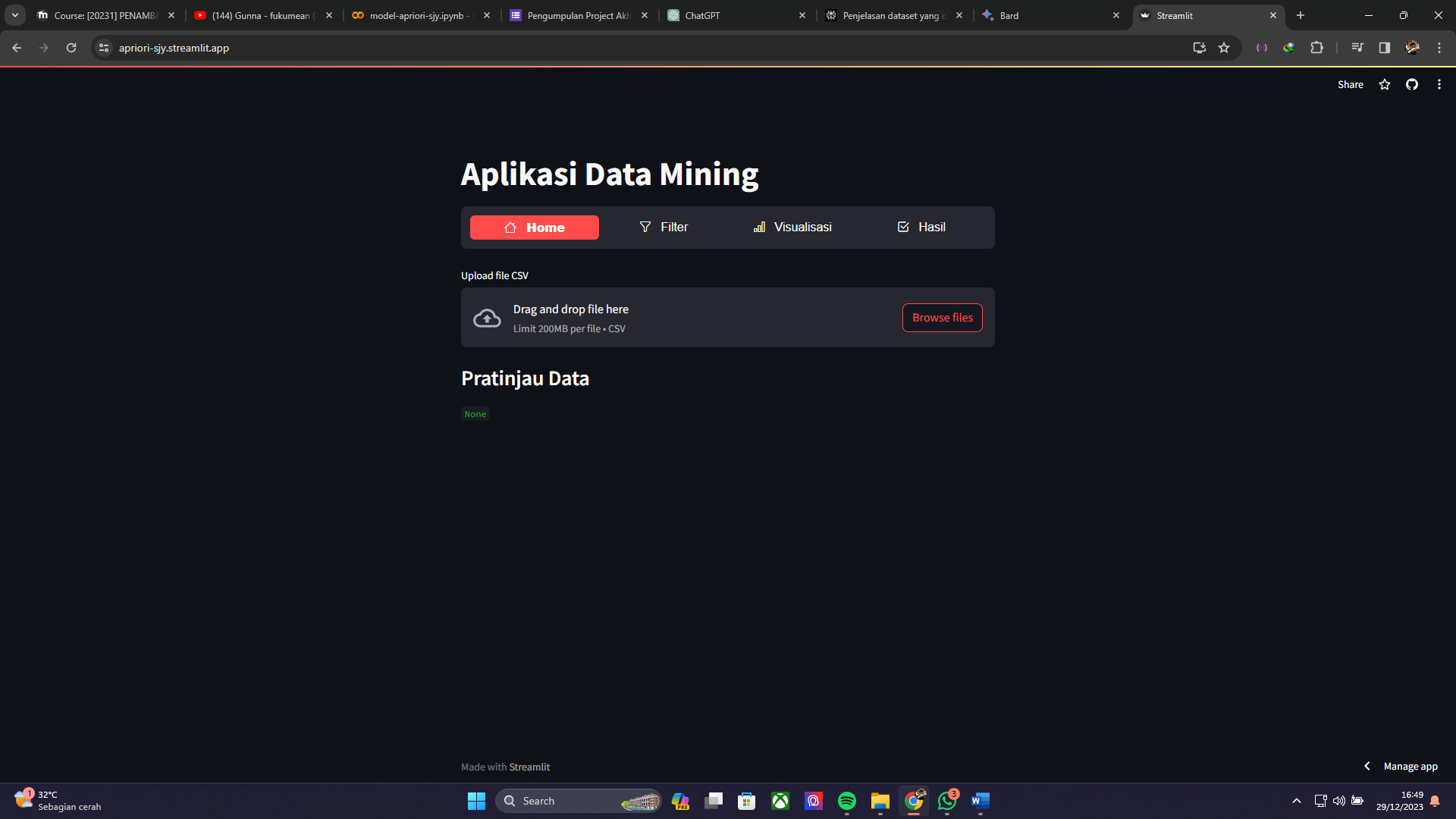
from mlxtend.frequent\_patterns import apriori, association\_rules

from datetime import datetime

import matplotlib.pyplot as plt

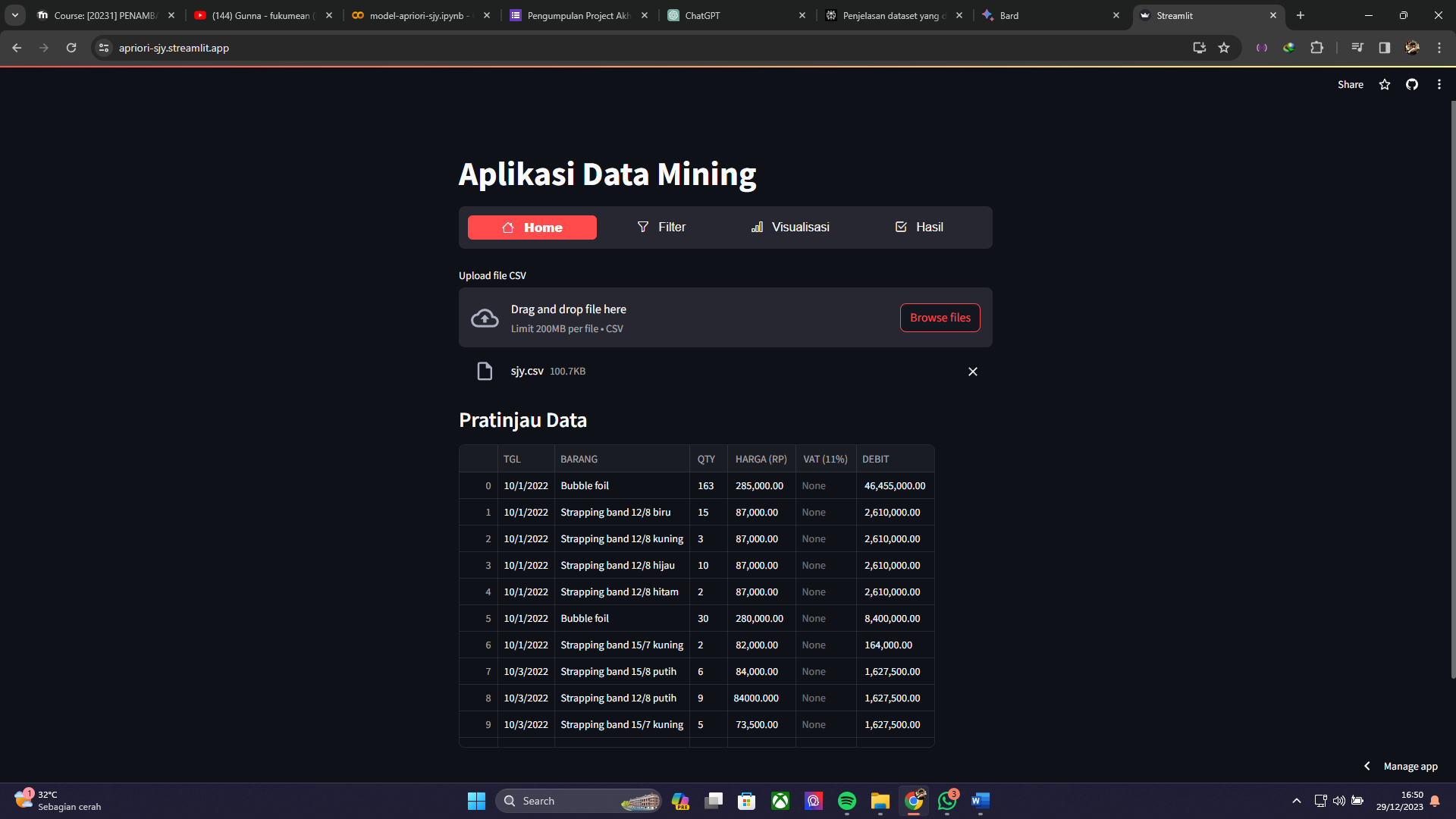
Melakukan import library yang dibutuhkan untuk membangun model.

### Input Data



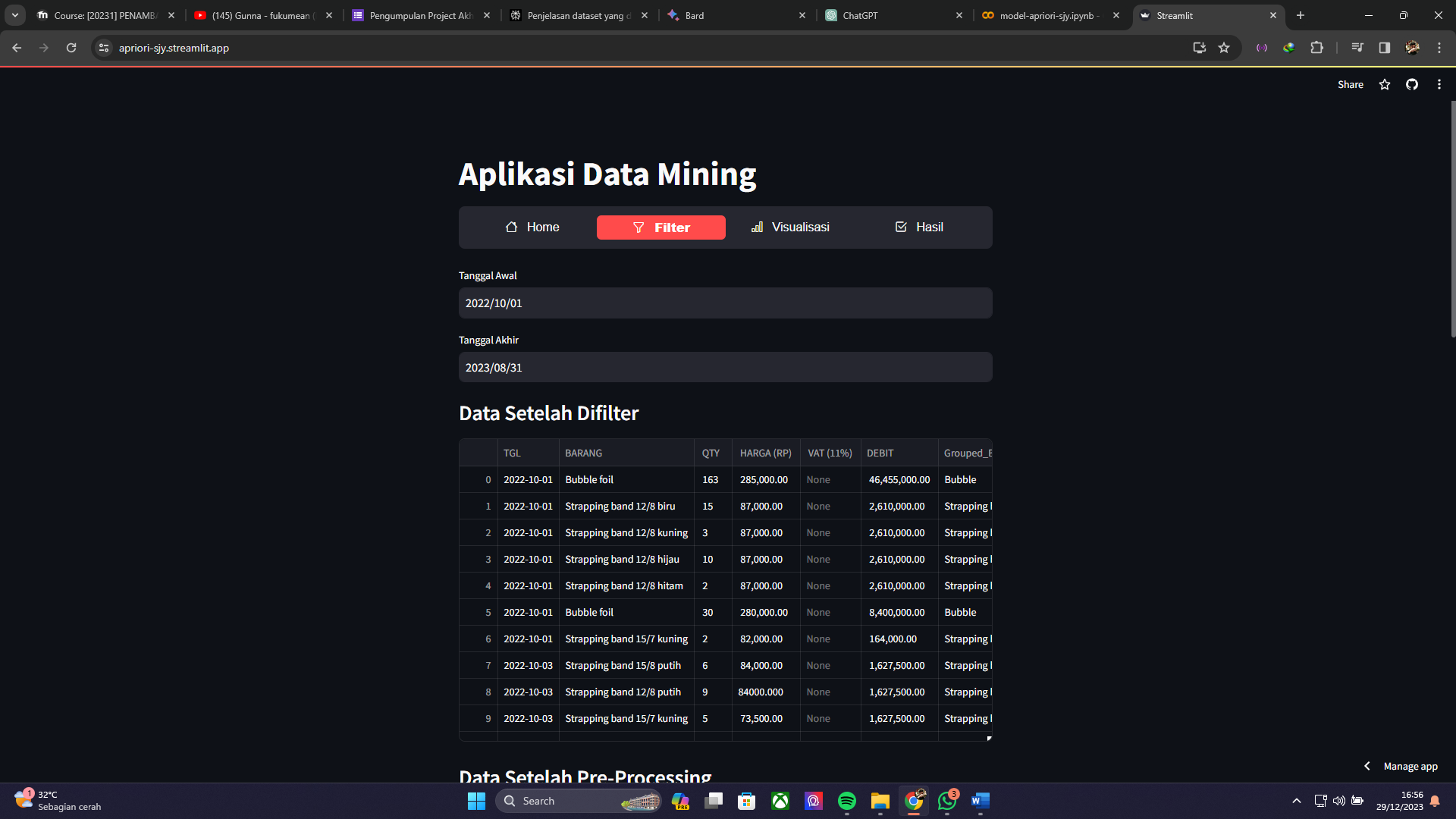
data = pd.read\_csv('sjy.csv')

Pada menu Home, klik browse file atau drop file data transaksi dalam format CSV. Setelah data diinput, fungsi dari Pandas akan membaca file CSV dan mengonversinya menjadi sebuah DataFrame.



Data tersebut akan ditampilkan pada menu home sebagai Pratinjau Data

### Filterisasi Data



filled\_data = data.fillna(method='ffill')

data = filled\_data

print(data.to\_string(index=False))

Pada menu Filter, data yang sebelumnya diinput akan diisi nilai-nilai kosongnya berdasarkan nilai dari baris sebelumnya

start\_date = datetime.strptime('2022-10-01', '%Y-%m-%d').date()

end\_date = datetime.strptime('2023-08-31', '%Y-%m-%d').date()

def filter\_data\_by\_date\_range(data, date\_column, start\_date, end\_date):

    # Konversi kolom tanggal dalam DataFrame menjadi objek datetime.date

    data[date\_column] = pd.to\_datetime(

        data[date\_column], errors='coerce').dt.date

    # Lakukan filter dengan perbandingan antara datetime.date

    filtered\_data = data[(data[date\_column] >= start\_date)

                         & (data[date\_column] <= end\_date)]

    filtered\_data['Grouped\_BARANG'] = filtered\_data['BARANG'].apply(

        group\_barang)

    filtered\_data['Proces\_Group\_BARANG'] = filtered\_data['BARANG'].apply(

        proces\_group\_barang)

    return filtered\_data

Setelah itu, menentukan range Tanggal Awal sampai Tanggal Akhir data yang ingin diproses.

# Grouping untuk visualisasi

def group\_barang(barang\_name):

    if 'Strapping band' in barang\_name:

        return 'Strapping band'

    if 'Strapping ban' in barang\_name:

        return 'Strapping band'

……

    else:

        return barang\_name

# Grouping untuk preprocessing

def proces\_group\_barang(barang\_name):

    if 'kuning' in barang\_name:

        return 'Strapping band kuning'

    if 'biru' in barang\_name:

        return 'Strapping band biru'

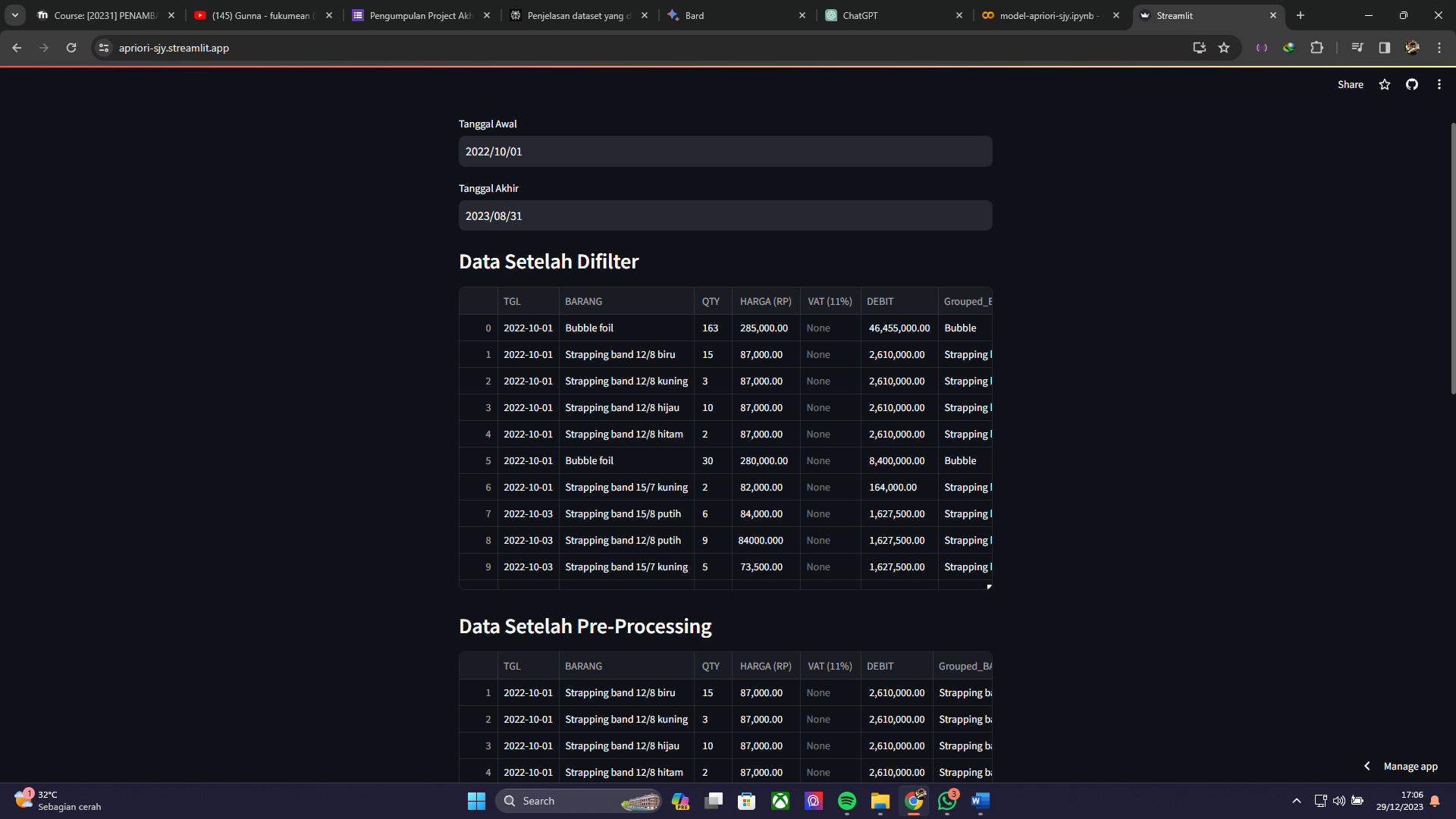
……

    else:

        return barang\_name

Selanjutnya membuat fungsi untuk mengelompokkan data barang menjadi 2 group yaitu:

1. Group\_barang : Berfungsi untuk bagian visualisasi
2. Proces\_group\_barang : Berfungsi untuk bagian pre-processing



Data kemudian akan ditampilkan dalam bentuk tabel

def preprocess\_data(data):

    # Menghitung jumlah kemunculan DEBIT dan TANGGAL

    debit\_date\_counts = data.groupby(

        ['DEBIT', 'TGL']).size().reset\_index(name='count')

    # Memilih DEBIT yang memiliki lebih dari 1 entri unik dengan TANGGAL yang berbeda

    valid\_debits = debit\_date\_counts['DEBIT'][debit\_date\_counts.groupby(

        'DEBIT')['TGL'].transform('nunique') > 1].unique()

    # Filter data berdasarkan DEBIT dan TANGGAL yang valid

    filtered\_data = data[data['DEBIT'].isin(valid\_debits)]

    # Melakukan pengelompokan hanya berdasarkan 'DEBIT' dan 'BARANG'

    transaksiSJY = (filtered\_data.groupby(['DEBIT', 'Proces\_Group\_BARANG'])

                    .size().unstack().reset\_index().fillna(0)

                    .set\_index('DEBIT'))

    transaksiSJY = transaksiSJY.apply(pd.to\_numeric, errors='coerce')

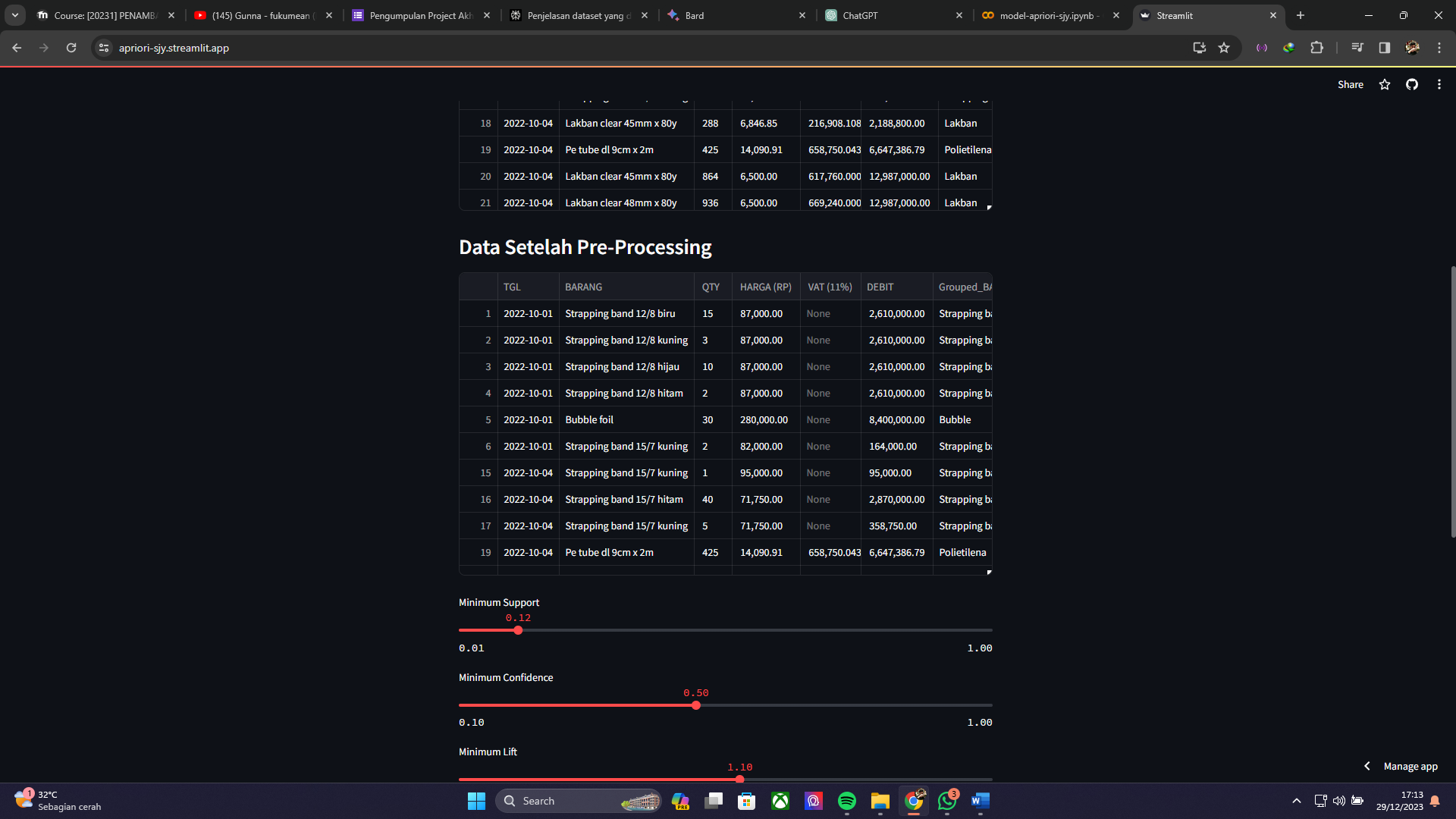
    transaksiSJYEncoded = transaksiSJY.applymap(lambda x: 1 if x >= 1 else 0)

    transaksiSJY = transaksiSJYEncoded

    transaksiSJY.fillna(0, inplace=True)

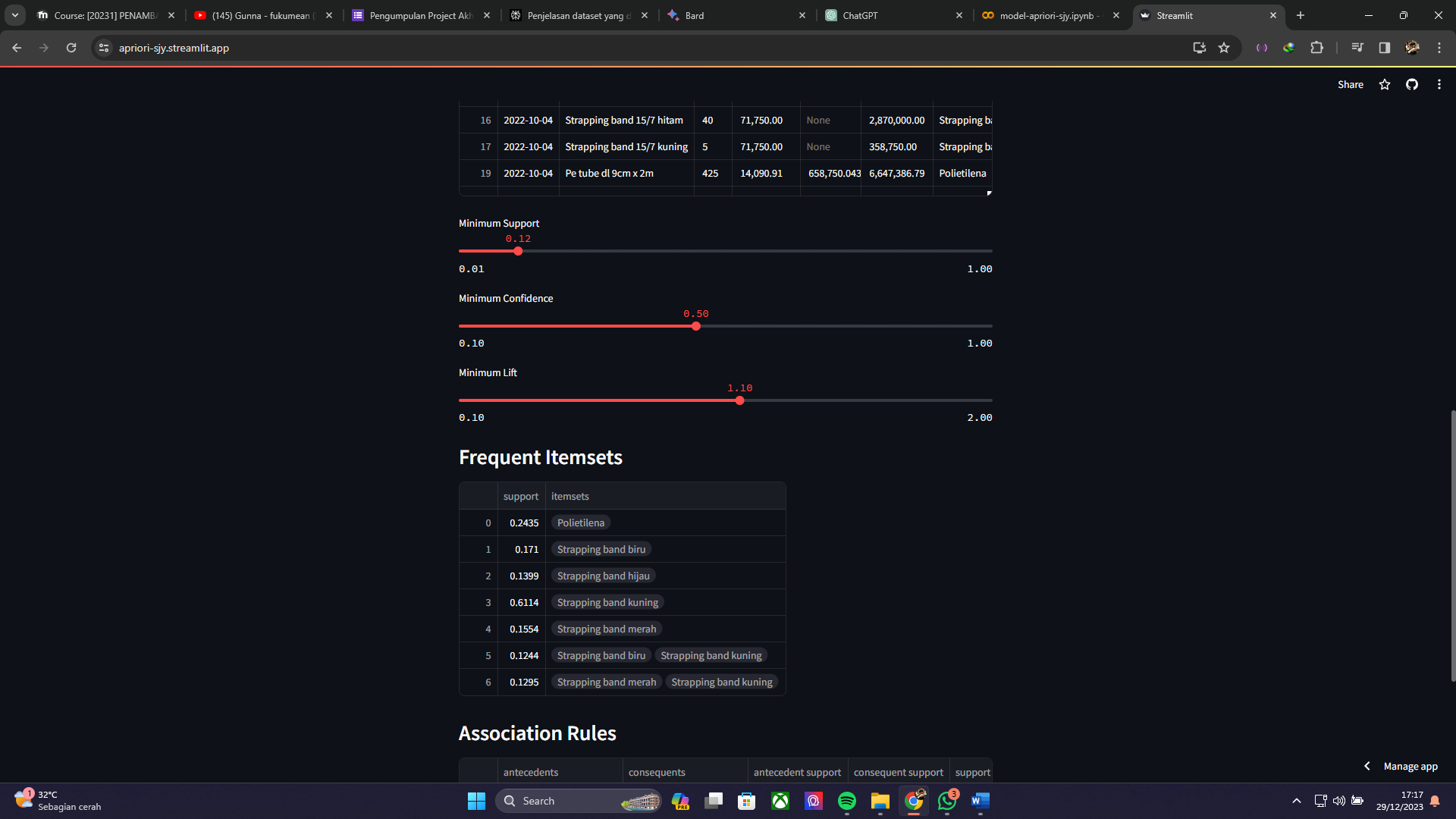
    return transaksiSJY

Setelah itu data akan masuk kedalam fungsi pre-processing dimana terjadi penggabungan kolom ‘TGL’ dengan ‘DEBIT’ sehingga jika terdapat debit dengan nilai yang sama namun memiliki tanggal yang berbeda akan dianggap sebagai transaksi yang berbeda. Kemudian menyeleksi data dengan mengambil transaksi yang didalamnya minimal terdapat 2 barang dalam sekali transaksi.



Data tersebut kemudian diubah ke bentuk numerik dan dikodekan sebagai 1 dan 0. Dimana 1 jika nilai = 1 atai nilai > 1 dan 0 dimana nilai = 0 atau nilai = NaN.

### Pembuatan Frequent Itemsets dan Association Rules



    # Run Apriori algorithm

    frq\_items = apriori(preprocessed\_data,

                        min\_support=0.12, use\_colnames=True)

    # Collect inferred rules into a dataframe

    rules = association\_rules(

        frq\_items, metric="lift", min\_threshold=0.5)

    rules = rules.sort\_values(

        ['confidence', 'lift'], ascending=[False, False])

    # Filter rules based on minimum lift

    rules = association\_rules(

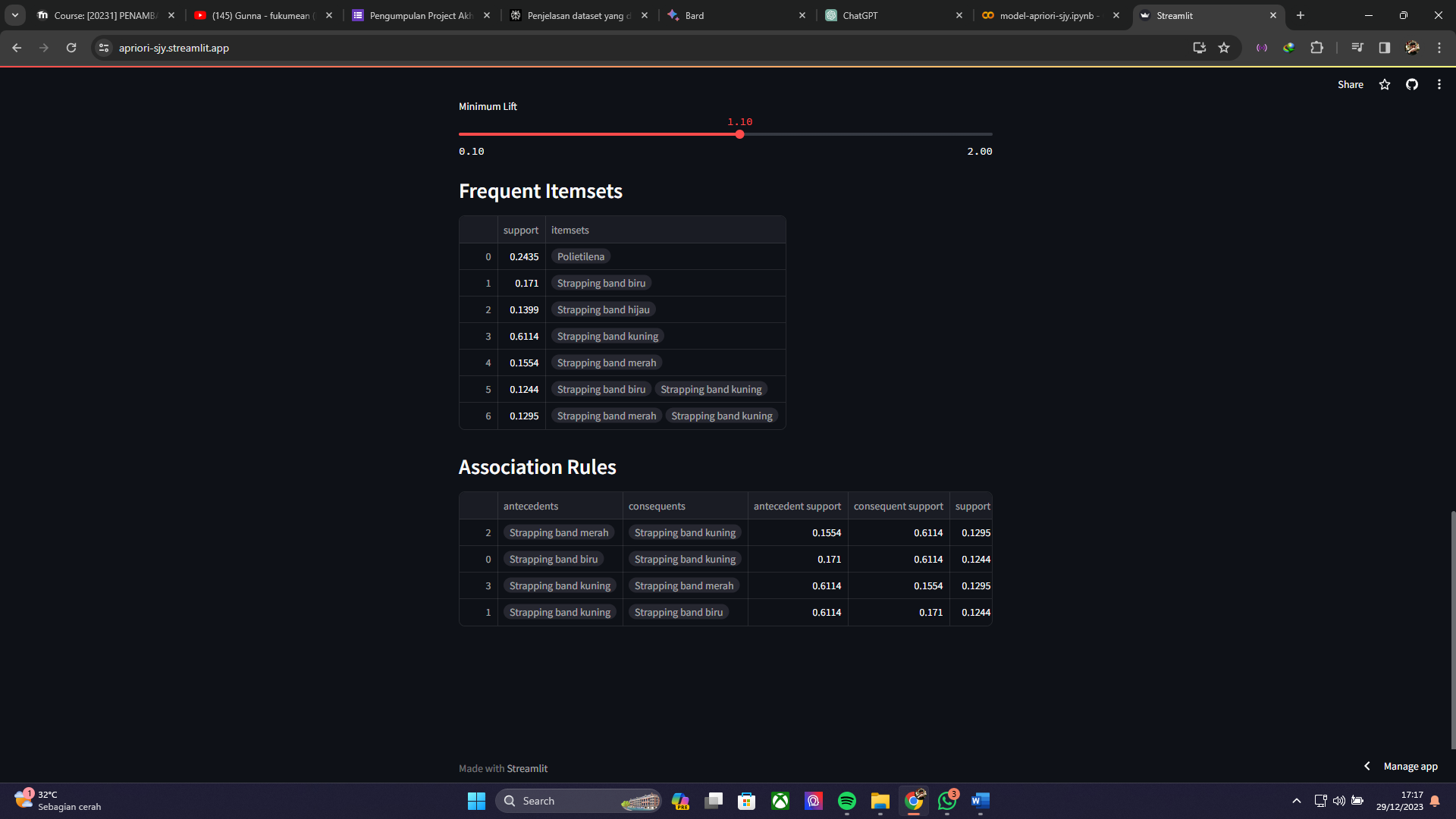
        frq\_items, metric="lift", min\_threshold=1.1)

    # Sort rules by confidence and lift in descending order

    rules = rules.sort\_values(

        ['confidence', 'lift'], ascending=[False, False])

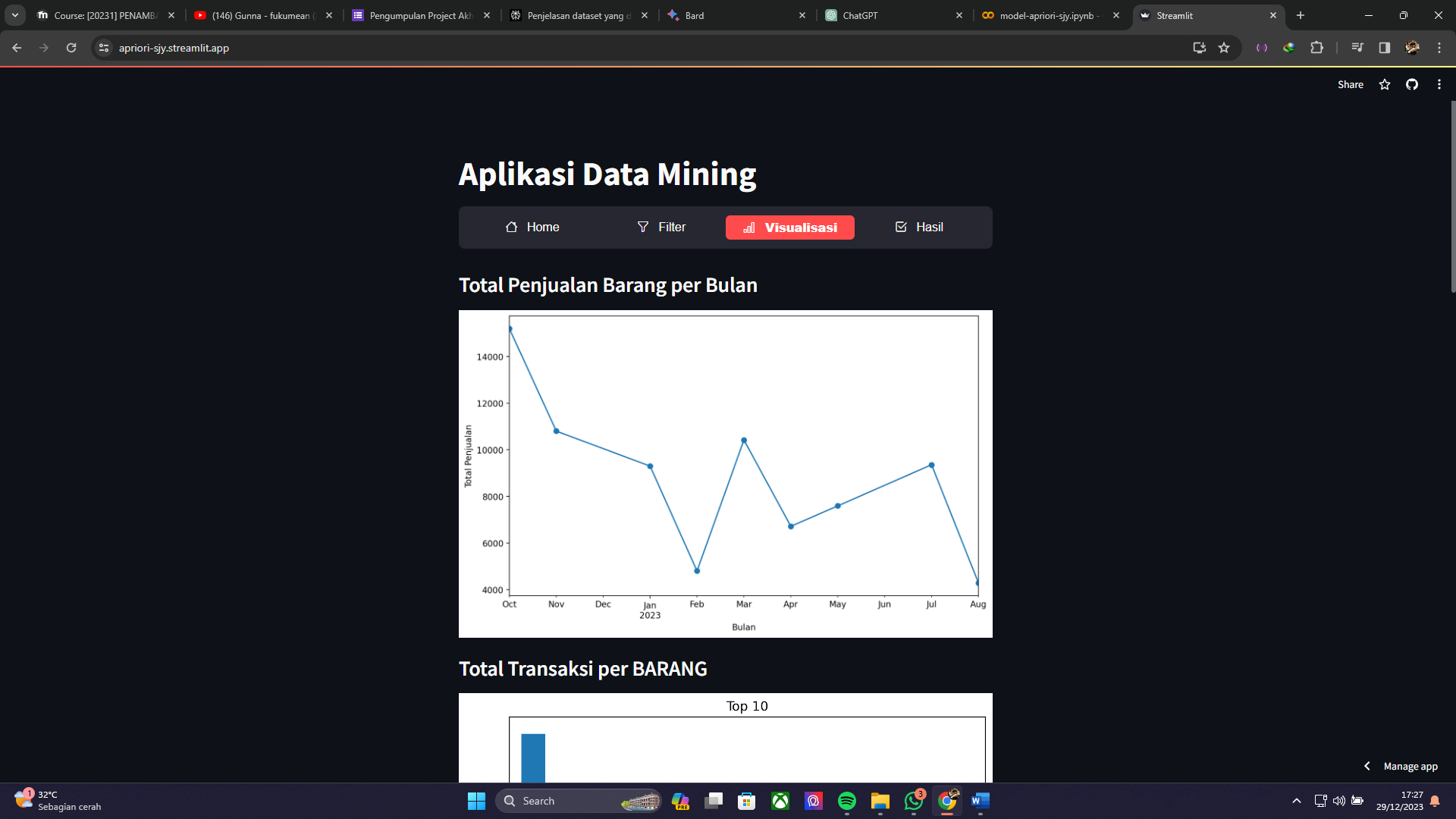
Proses ini menggunakan group dari proces\_group\_barang. Selanjutnya menentukan nilai Minimum Support (0.12), Minimum Confidence(0.5), dan Minimum Lift (1.1) untuk membuat Association Rules. Rules yang sudah terbentuk akan diurutkan dari nilai terbesar hingga terkecil.



Terbentuk 4 Rules yaitu :

1. Strapping Band Merah dengan Strapping Band Kuning dengan nilai support 0.129534, nilai confidence 0.833333, dan nilai lift 1.362994.
2. Strapping Band Biru dengan Strapping Band Kuning dengan nilai support 0.124352, nilai confidence 0.727273, dan nilai lift 1.189522.
3. Strapping Band Kuning dengan Strapping Band Merah dengan nilai support 0.129534, nilai confidence 0.211864, dan nilai lift 1.362994.
4. Strapping Band Kuning dengan Strapping Band Biru dengan nilai support 0.124352, nilai confidence 0.203390, dan nilai lift 1.189522.

### Visualisasi Data



def plot\_data\_preprocessing(filtered\_data):

    # Mengonversi kolom TGL menjadi tipe data datetime

    filtered\_data['TGL'] = pd.to\_datetime(

        filtered\_data['TGL'], format='%Y/%m/%d')

    # Mengonversi kolom QTY ke tipe data numerik dan mengganti NaN dengan 0

    filtered\_data[' QTY '] = pd.to\_numeric(

        filtered\_data[' QTY '], errors='coerce').fillna(0)

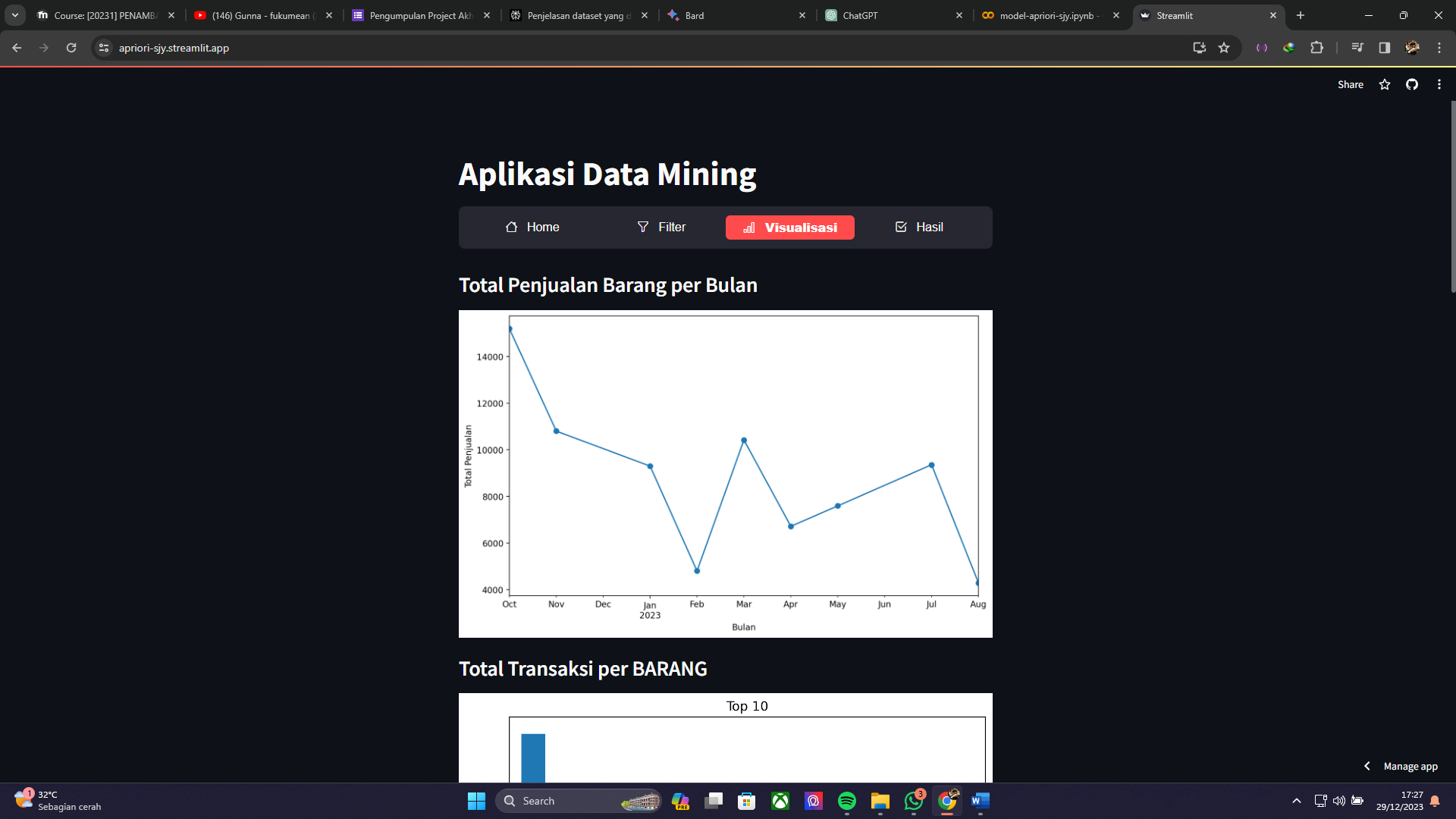
    # Membuat kolom baru 'Bulan' untuk menyimpan informasi bulan dari tanggal transaksi

    filtered\_data['Bulan'] = filtered\_data['TGL'].dt.to\_period('M')

    # Menghitung total penjualan (QTY) per bulan

    sales\_per\_month = filtered\_data.groupby('Bulan')[' QTY '].sum()

Pada menu Visualisasi, data yang sudah melalui menu Filter akan ditampilkan dalam bentuk grafik (Line Chart, Bar Chart, Pie Chart) menggunakan fungsi dari plot\_data\_preprocessing



    # Visualisasi line chart untuk total penjualan per bulan

    plt.figure(figsize=(10, 6))

    sales\_per\_month.plot(kind='line', marker='o', linestyle='-')

    plt.xlabel('Bulan')

    plt.ylabel('Total Penjualan')

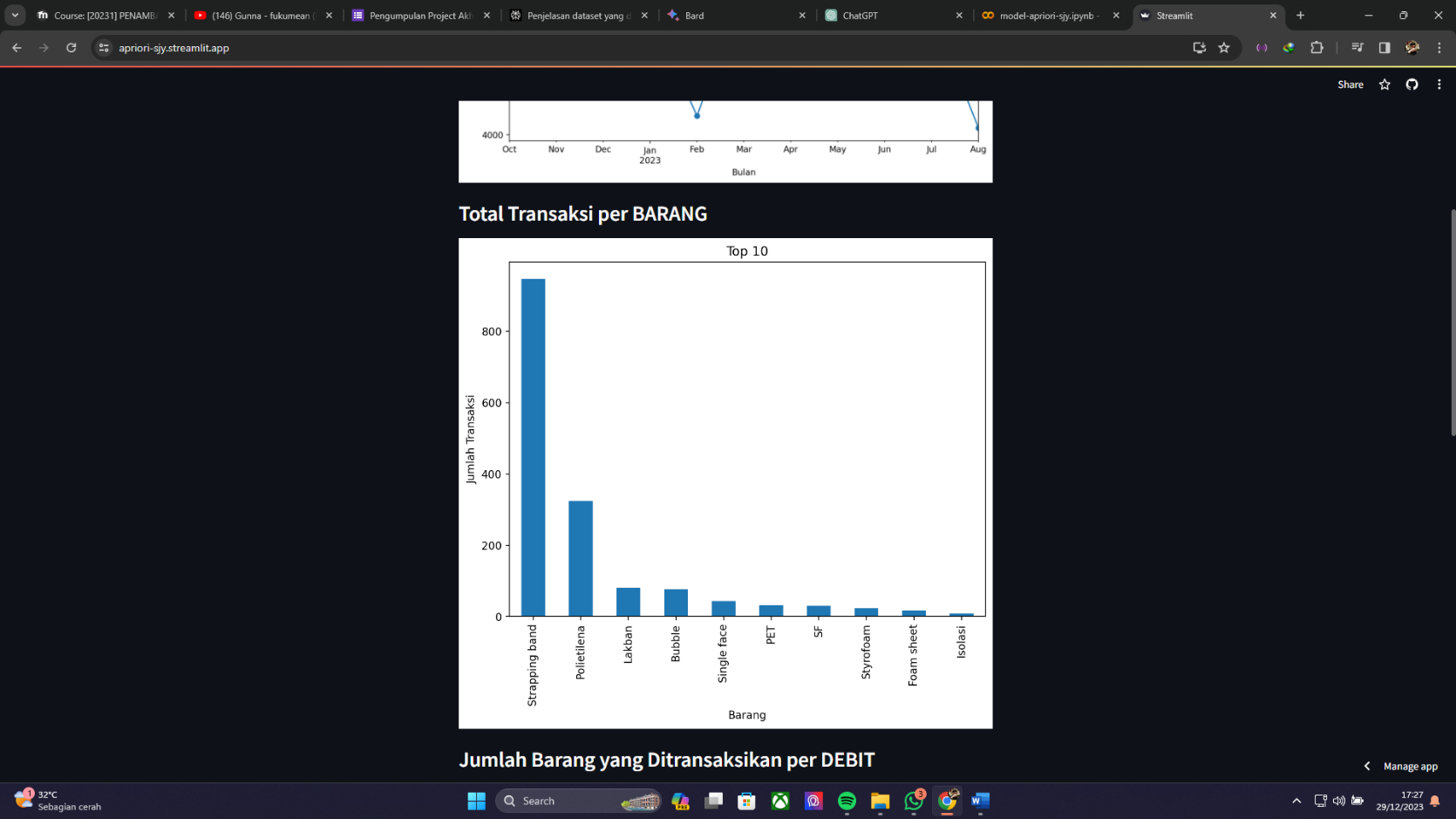
    plt.title('Total Penjualan Barang per Bulan')

    plt.tight\_layout()

    plt.show()

    plt.subplots\_adjust(hspace=0.5)

Grafik ini menampilkan berapa banyak jumlah barang yang dijual per bulannya.



    # Visualisasi grafik untuk jumlah transaksi per BARANG (hanya 10 data teratas)

    plt.figure(figsize=(8, 6))

    filtered\_data['Grouped\_BARANG'].value\_counts().nlargest(

        10).plot(kind='bar')

    plt.xlabel('Barang')

    plt.ylabel('Jumlah Transaksi')

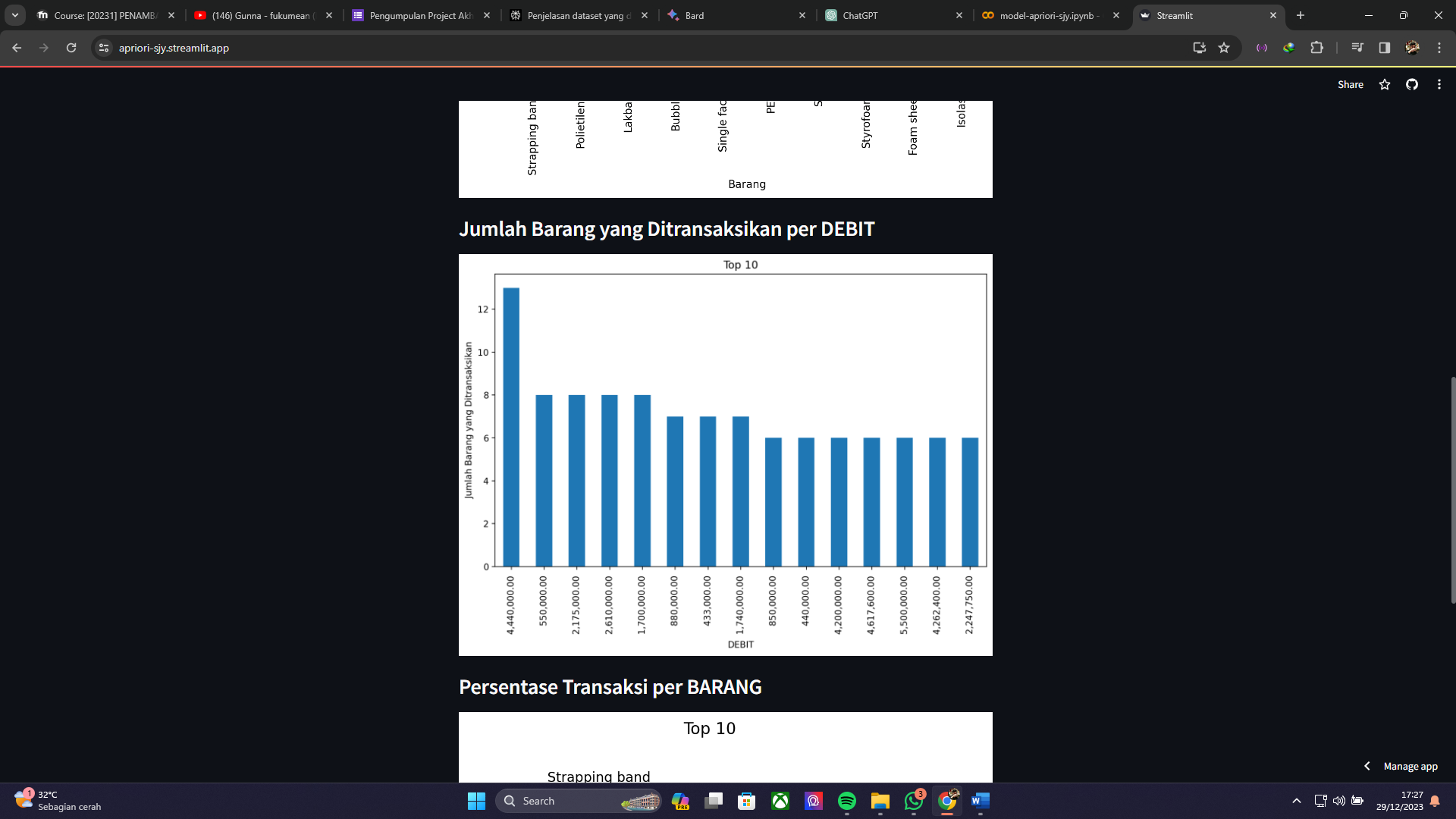
    plt.title('Top 10')

    plt.tight\_layout()

    plt.show()

    plt.subplots\_adjust(hspace=0.5)

Grafik ini menampilkan seberapa sering barang tersebut muncul dalam terjadinya transaksi.



    # Menghitung jumlah barang yang ditransaksikan per DEBIT

    barang\_per\_debit = filtered\_data.groupby(

        'DEBIT')['BARANG'].nunique().sort\_values(ascending=False)

    # Visualisasi line chart untuk jumlah barang yang ditransaksikan per DEBIT (hanya 10 data teratas)

    plt.figure(figsize=(10, 6))

    barang\_per\_debit.head(15).plot(kind='bar')

    plt.xlabel('DEBIT')

    plt.ylabel('Jumlah Barang yang Ditransaksikan')

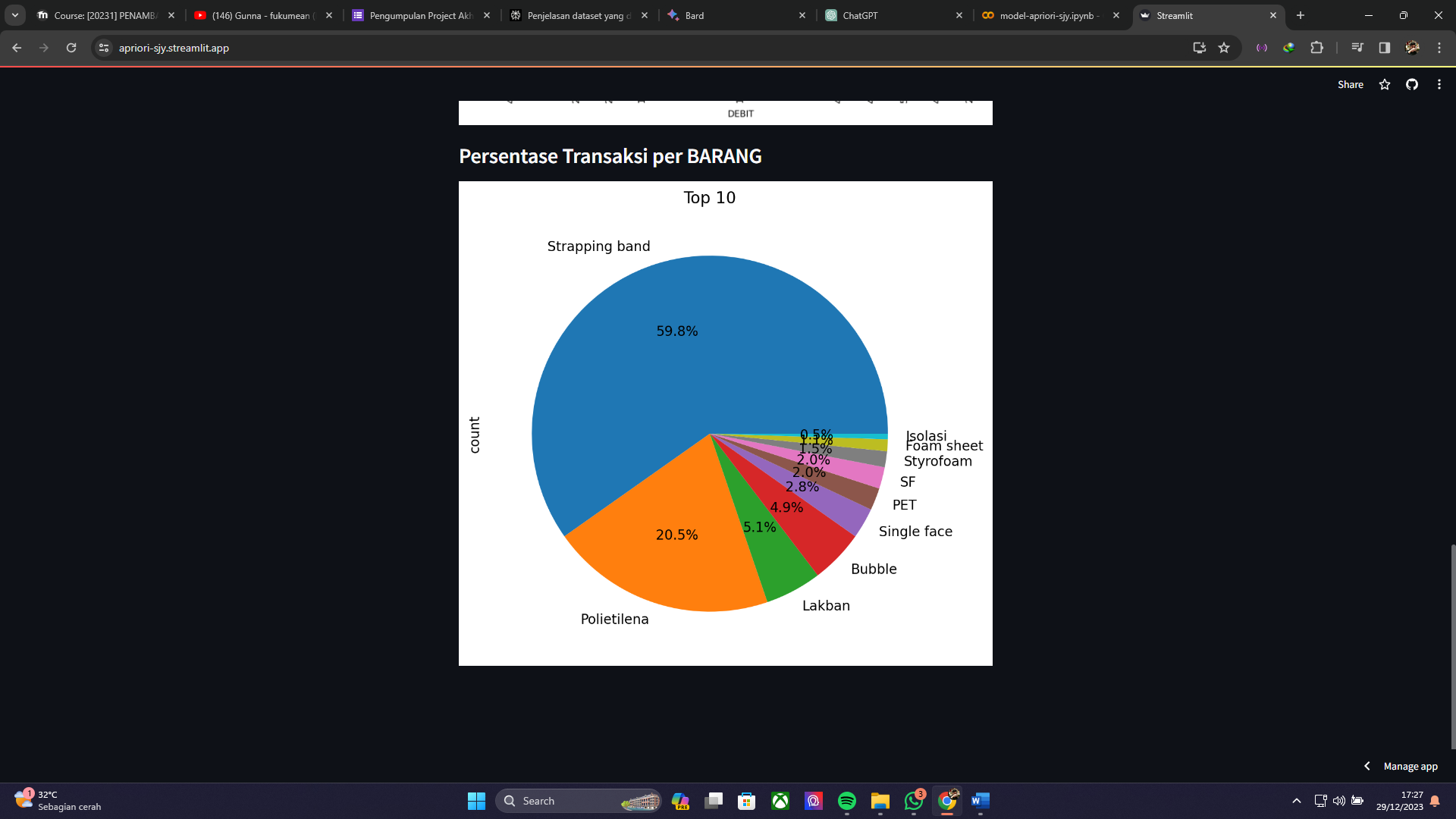
    plt.title('Top 10')

    plt.tight\_layout()

    plt.show()

    plt.subplots\_adjust(hspace=0.5)

Grafik ini menampilkan berapa banyak jumlah barang yang biasanya ditransaksikan dalam sekali pembayaran debit



    # Visualisasi pie chart untuk jumlah transaksi per BARANG (hanya 10 data teratas)

    plt.figure(figsize=(8, 6))

    filtered\_data['Grouped\_BARANG'].value\_counts().nlargest(

        10).plot(kind='pie', autopct='%1.1f%%')

    plt.title('Top 10')

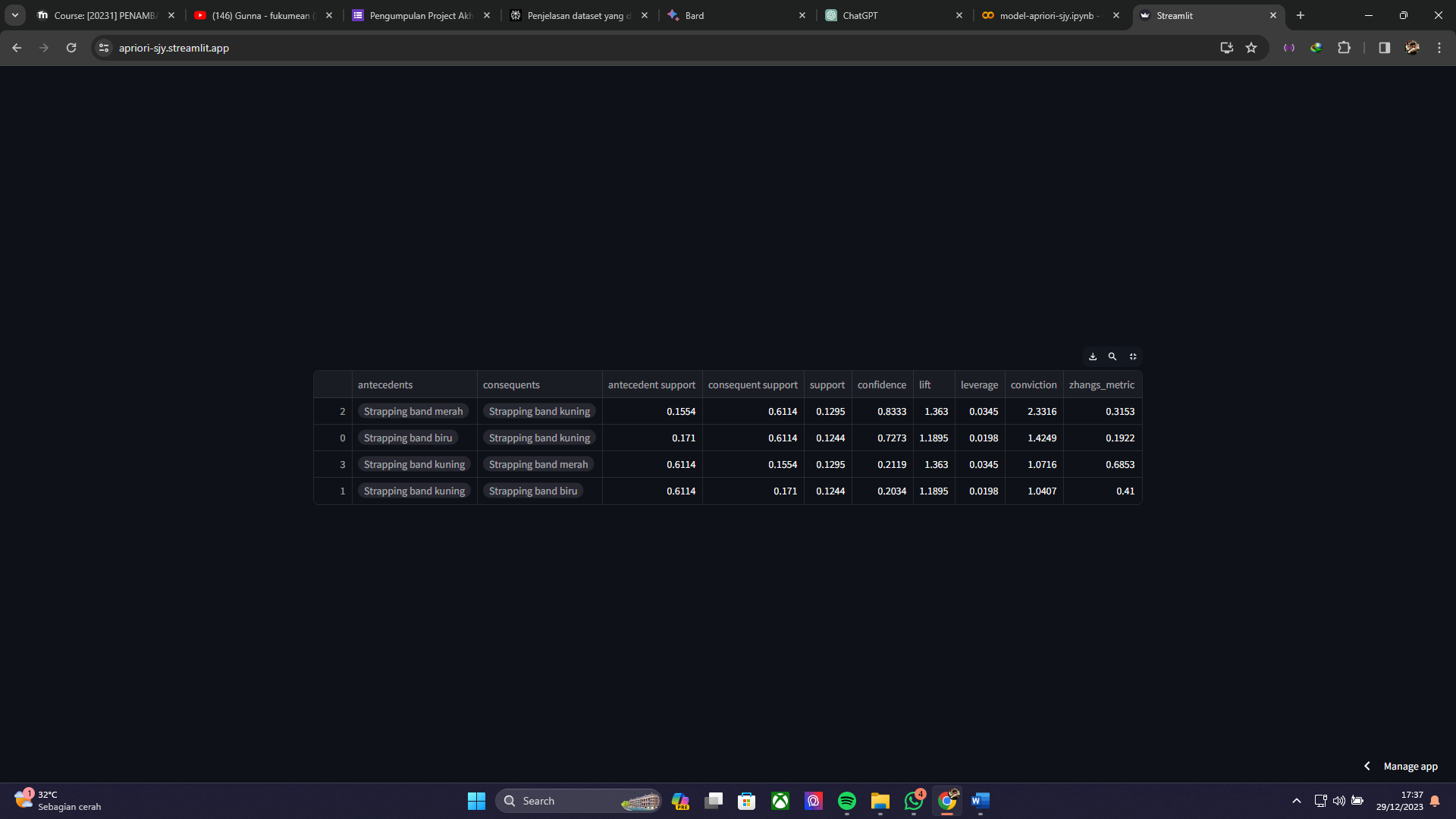
    plt.tight\_layout()

    plt.show()

Grafik ini menampilkan berapa persen barang tersebut menguasai total transaksi

## Performa Model / Uji Performa Model

Dengan menggunakan Algoritma Apriori, data yang sudah melewati proses filterisasi akan diproses lalu dibentuk menjadi Association Rules.



Dengan menggunakan nilai Minimum Support (0.12), Minimum Confidence(0.5), dan Minimum Lift (1.1) menghasilkan 4 Rules yaitu :

1. Strapping Band Merah dengan Strapping Band Kuning dengan nilai support 0.129534, nilai confidence 0.833333, dan nilai lift 1.362994.
2. Strapping Band Biru dengan Strapping Band Kuning dengan nilai support 0.124352, nilai confidence 0.727273, dan nilai lift 1.189522.
3. Strapping Band Kuning dengan Strapping Band Merah dengan nilai support 0.129534, nilai confidence 0.211864, dan nilai lift 1.362994.
4. Strapping Band Kuning dengan Strapping Band Biru dengan nilai support 0.124352, nilai confidence 0.203390, dan nilai lift 1.189522.

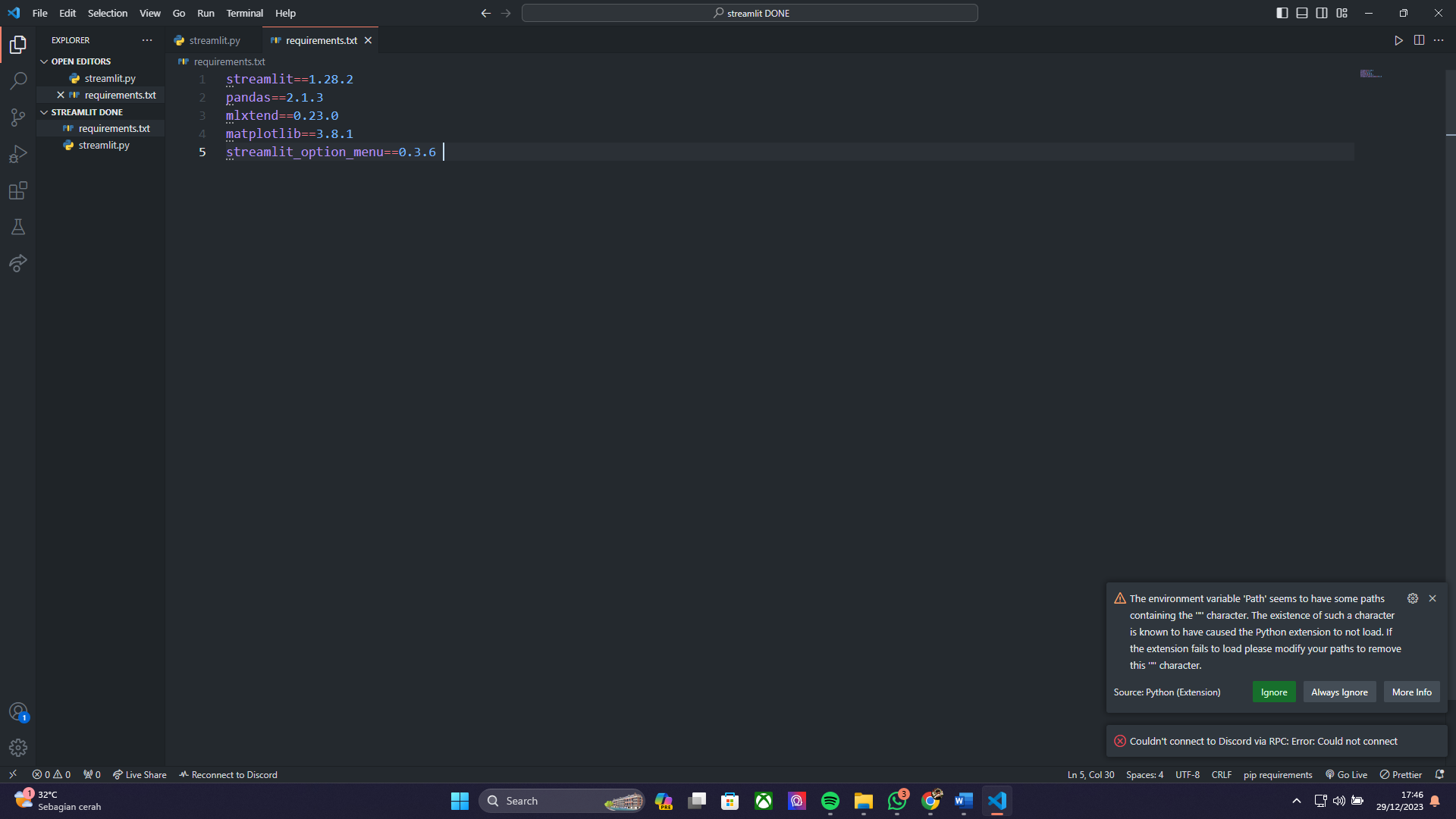
Rules yang terbentuk diurutkan dari yang mempunyai nilai terbesar hingga terkecil

## Proses Deployment

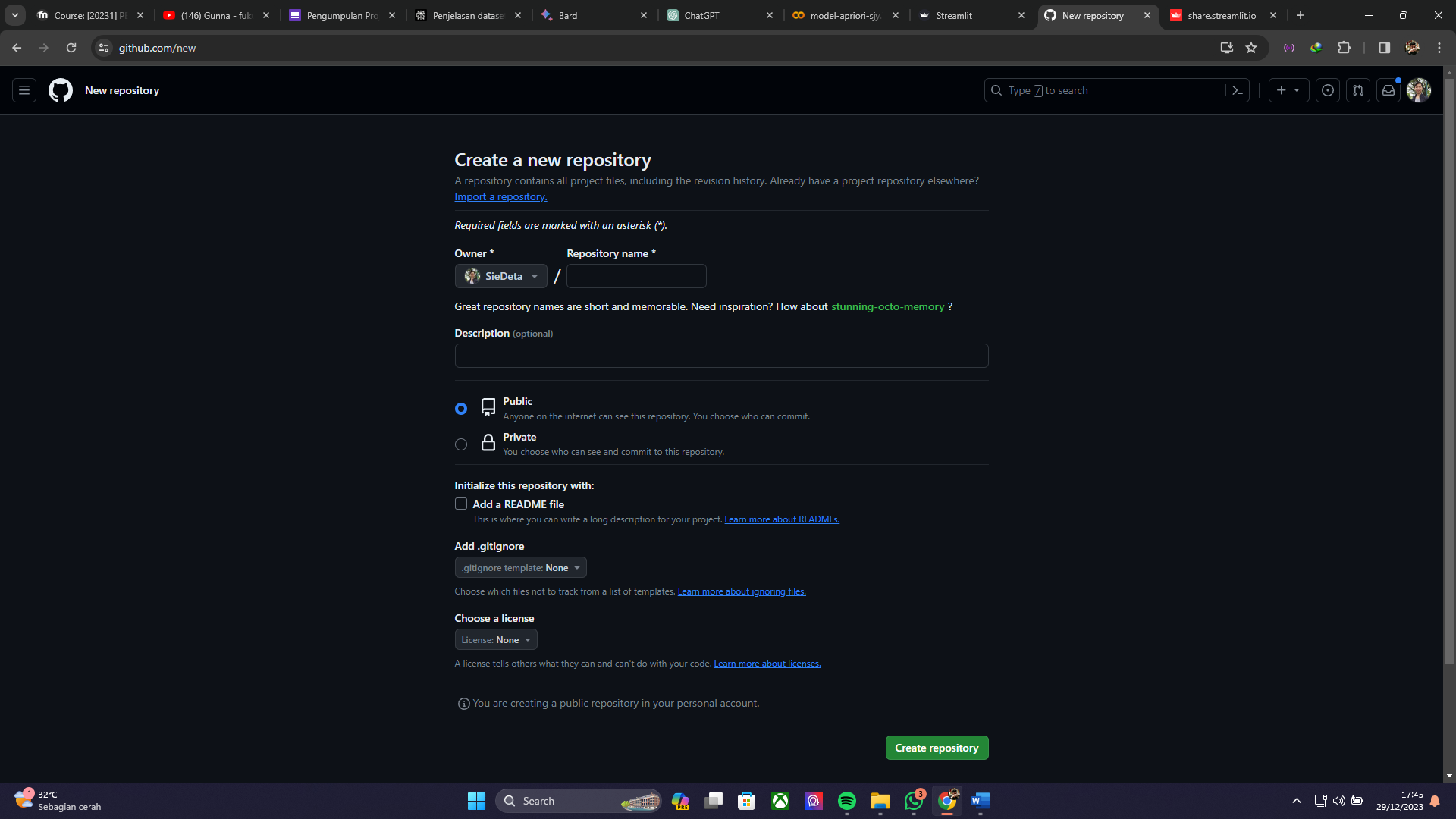
Aplikasi ini sudah di deploy menggunakan Streamlit Share (<https://share.streamlit.io/>) dan dapat diakses di URL <https://apriori-sjy.streamlit.app/> . Berikut langkah-langkah proses deployment yang dilakukan :

1. Membuat file requirements.txt

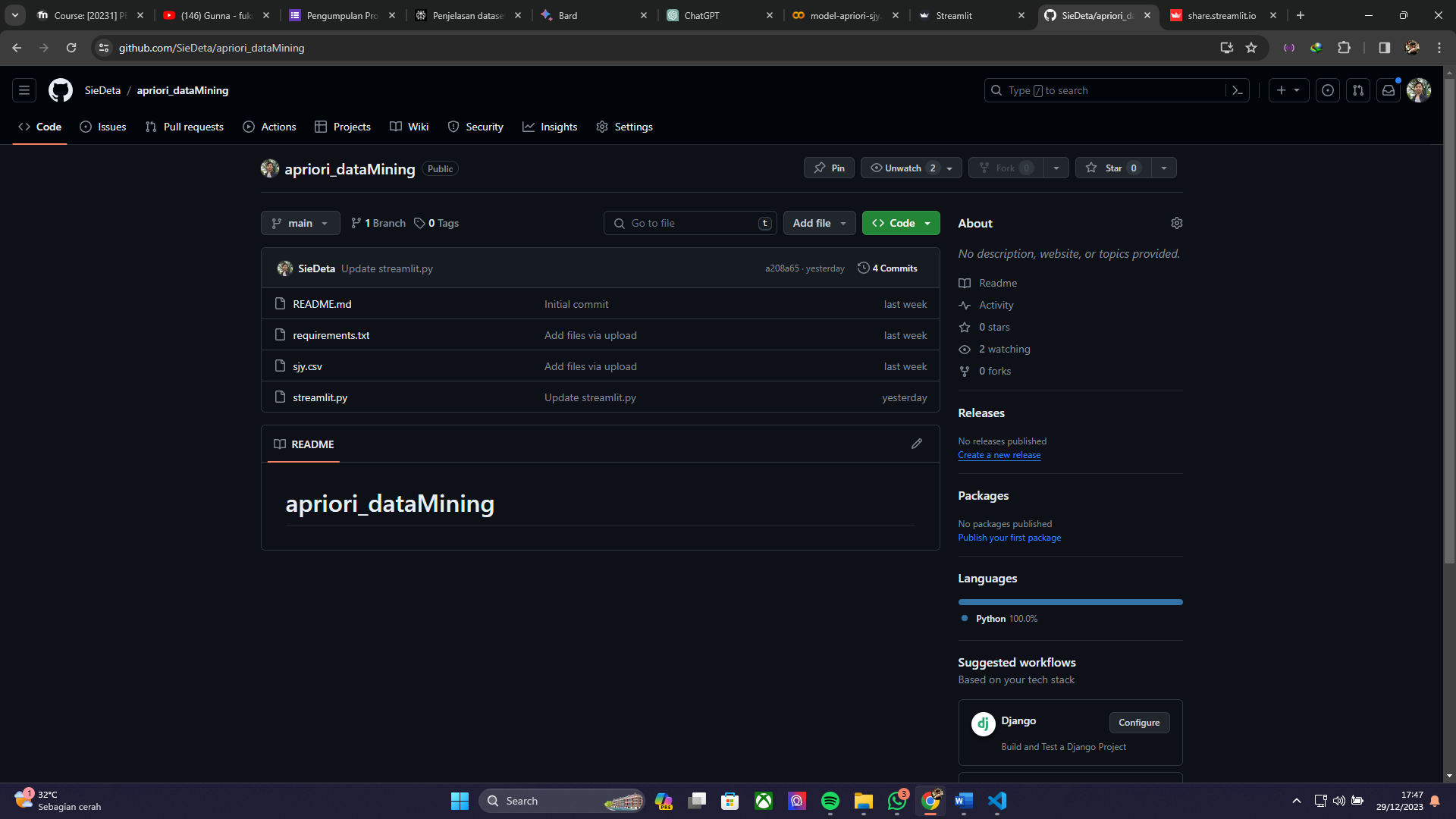
Fungsinya untuk membantu Instalasi Paket dalam proses deployment



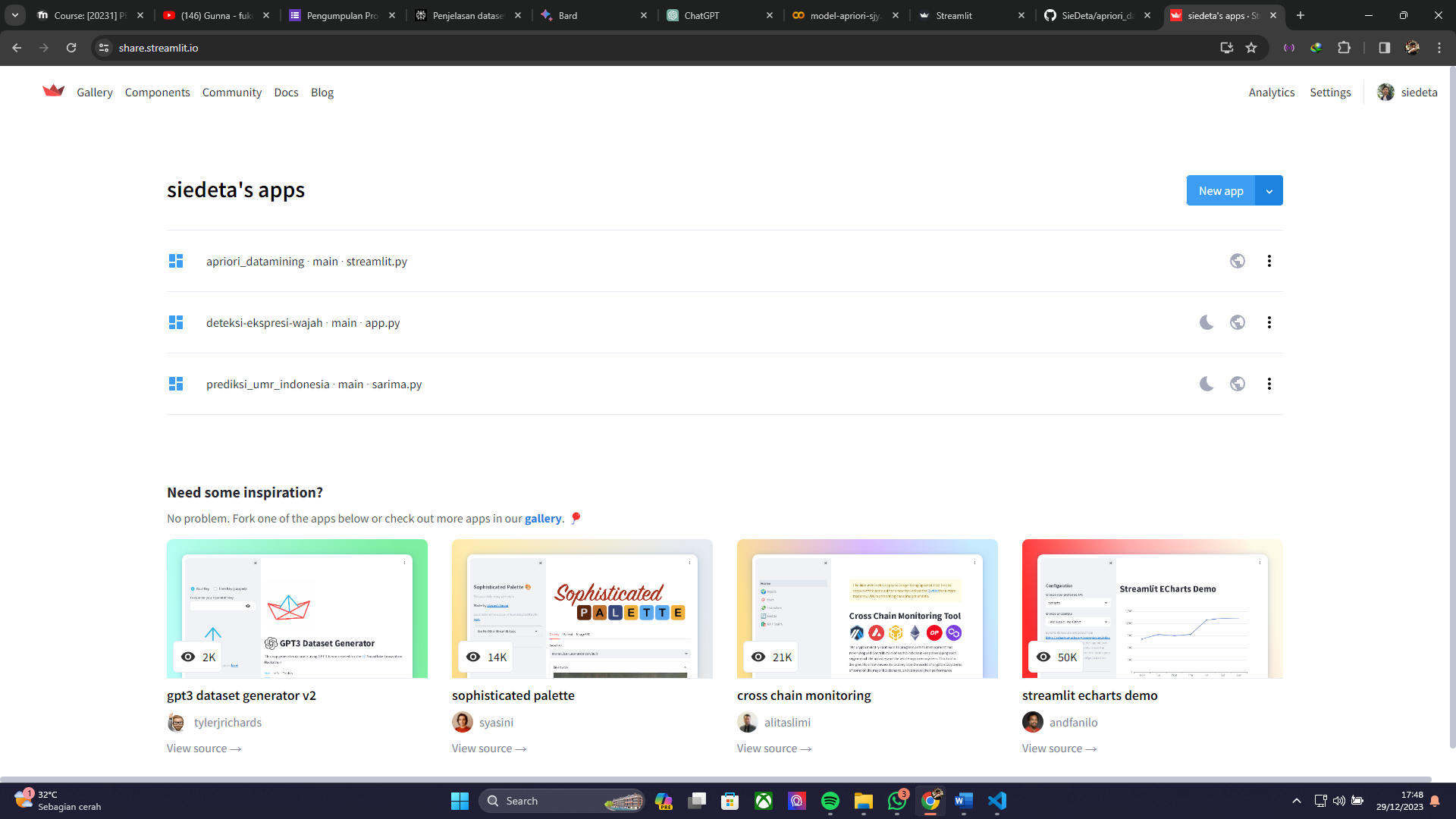
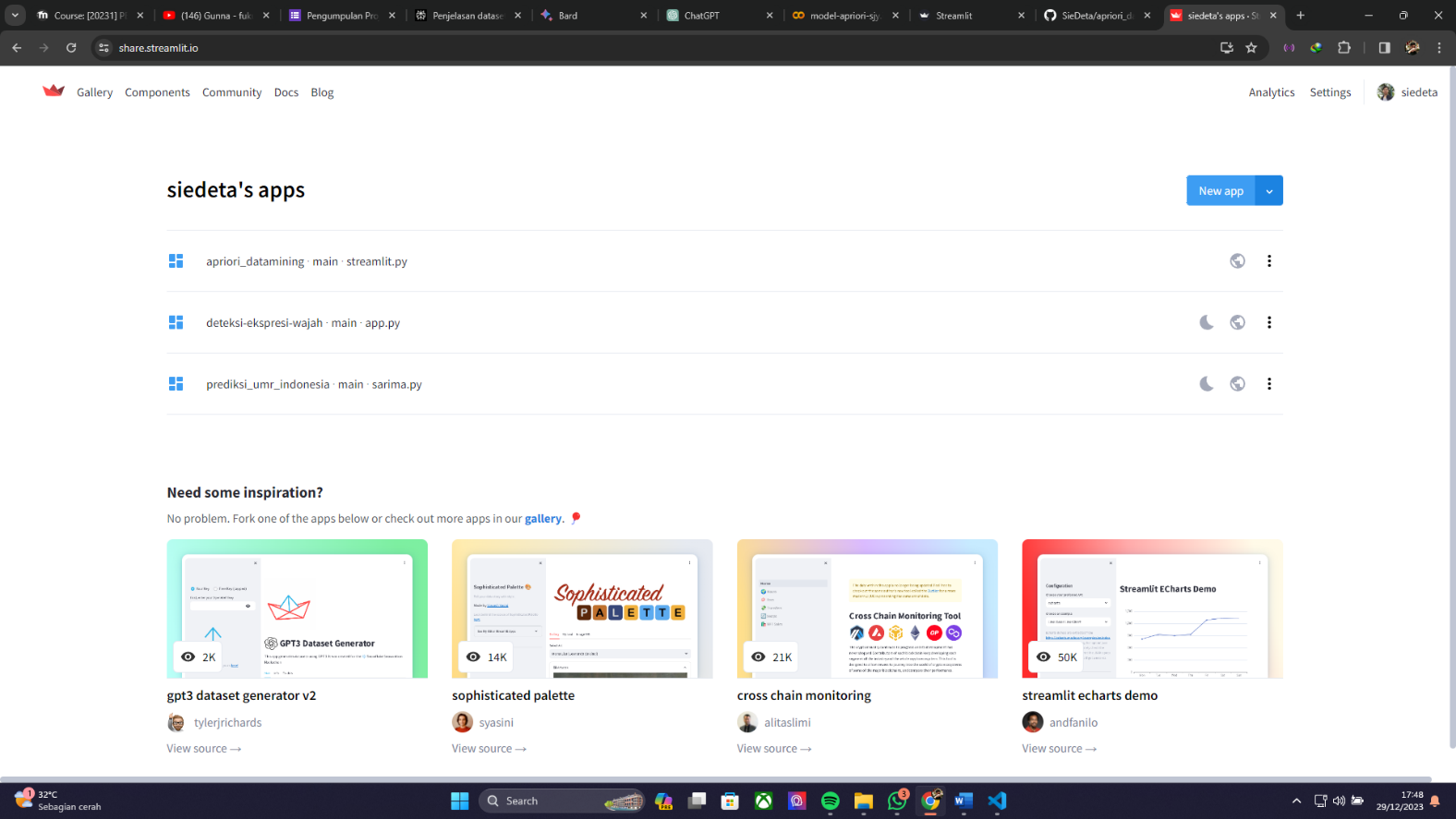
1. Membuat repositori baru di Github



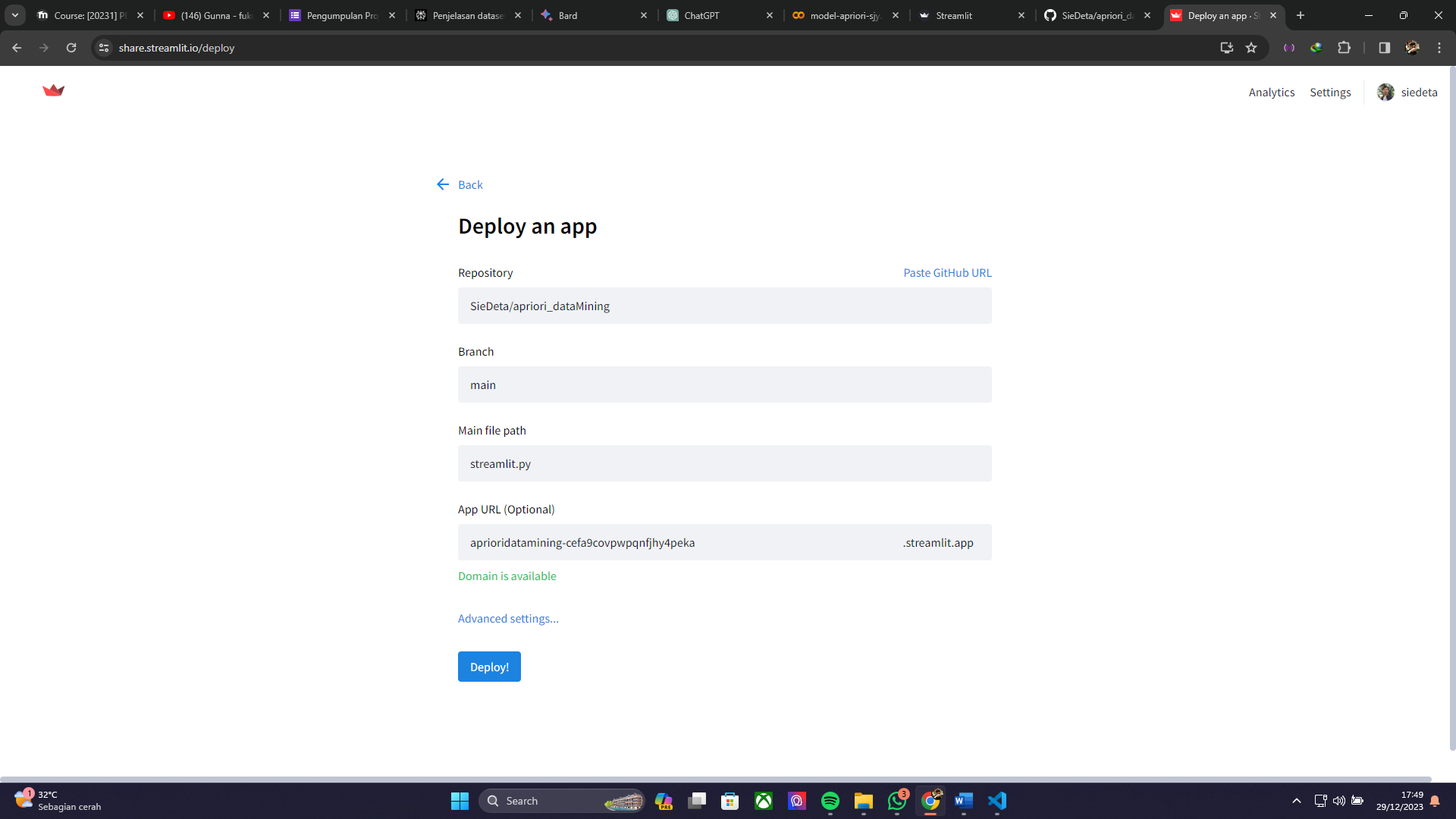
1. Upload semua file yang dibutuhkan ke dalam Github



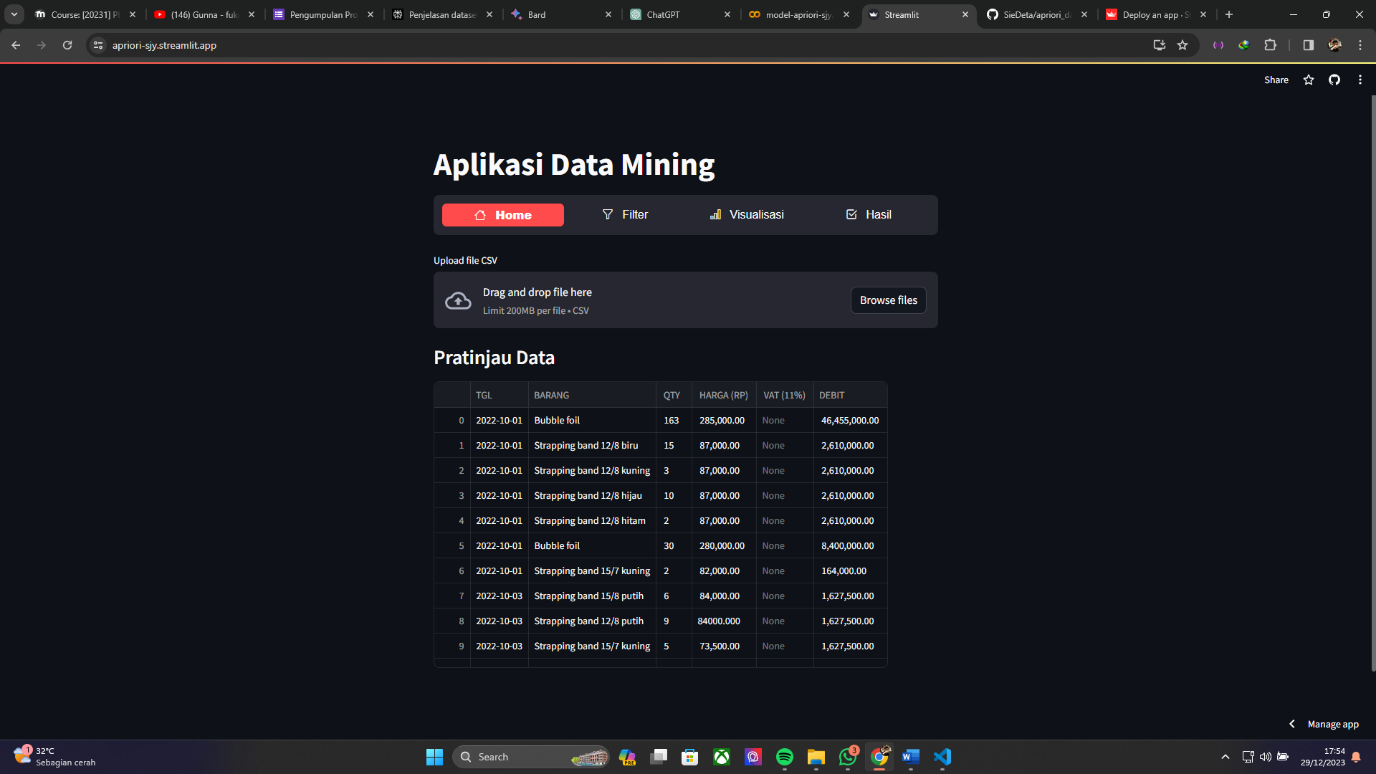
1. Masuk ke halaman <https://share.streamlit.io/> lalu klik “New APP”



1. Isi semua informasi yang dibutuhkan untuk proses deploy



1. Klik deploy lalu tunggu sampai proses deploy selesai. Setelah deploy selesai, aplikasi tersebut dapat diakses melalui domain yang dicustom kapan saja.



## Kesimpulan dan Rekomendasi

Hasil kesimpulan dan rekomendasi dapat diakses melalui menu Hasil berupa Pola Transaksi dan Rekomendasi Promo.

    print('Pola Transaksi:')

    sorted\_rules\_recommendation = rules.sort\_values(

        ['confidence', 'lift'], ascending=[False, False]).reset\_index(drop=True)

    for index, row in sorted\_rules\_recommendation.iterrows():

        antecedents = row['antecedents']

        consequents = row['consequents']

        antecedents\_str = ', '.join(str(item) for item in antecedents)

        consequents\_str = ', '.join(str(item) for item in consequents)

        recommendation\_text = f"{index + 1}. Jika pelanggan membeli {antecedents\_str}, maka biasanya membeli {consequents\_str}."

        print(recommendation\_text)

Pola Transaksi mengambil rules yang sudah dibentuk lalu ditampilkan dengan kata-kata yang lebih mudah dipahami.

    print('\nHasil Rekomendasi:')

    seen\_pairs = set()

    sorted\_rules\_knowledge = rules.sort\_values(['confidence', 'lift'], ascending=[False, False]).reset\_index(drop=True)

    for index, row in sorted\_rules\_knowledge.iterrows():

        antecedents = row['antecedents']

        consequents = row['consequents']

        pair = tuple(sorted([tuple(antecedents), tuple(consequents)]))

        if pair not in seen\_pairs:

            antecedents\_str = ', '.join(str(item) for item in antecedents)

            consequents\_str = ', '.join(str(item) for item in consequents)

            knowledge\_text = f"{len(seen\_pairs) + 1}. Membuat promo menggunakan pasangan barang {antecedents\_str} dengan barang {consequents\_str}."

            print(knowledge\_text)

            seen\_pairs.add(pair)

Hasil Rekomendasi menyeleksi hasil dari Pola Transaksi, jika terdapat nama barang yang sama tetapi dalam pola yang berbeda, maka hanya akan mengambil salah satu dari pola tersebut.

Hasil Kesimpulan dan Rekomendasi dari model adalah :

