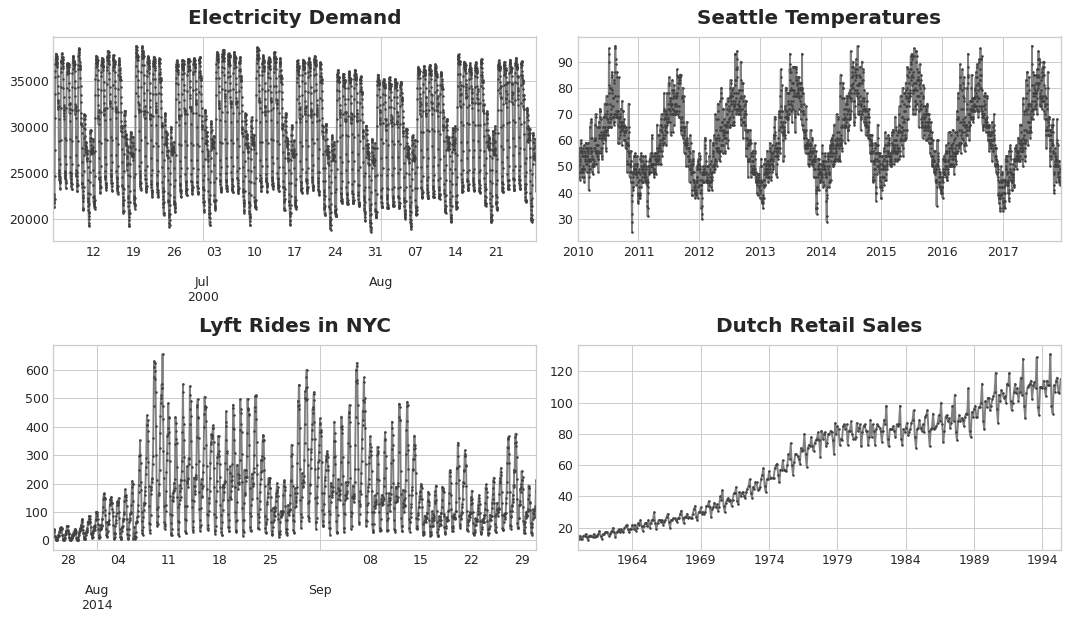
What is Seasonality?

Bir zaman serisi, serinin ortalamasında düzenli, periyodik bir değişim olduğunda **mevsimsellik** gösterir deriz. Mevsimsel değişiklikler genellikle saat ve takvimi takip eder; bir gün, bir hafta veya bir yıl boyunca tekrarlar yaygındır. Mevsimsellik, çoğunlukla doğal dünyanın günler ve yıllar üzerindeki döngüleri veya tarih ve saatlerle ilgili sosyal davranış alışkanlıkları tarafından yönlendirilir.



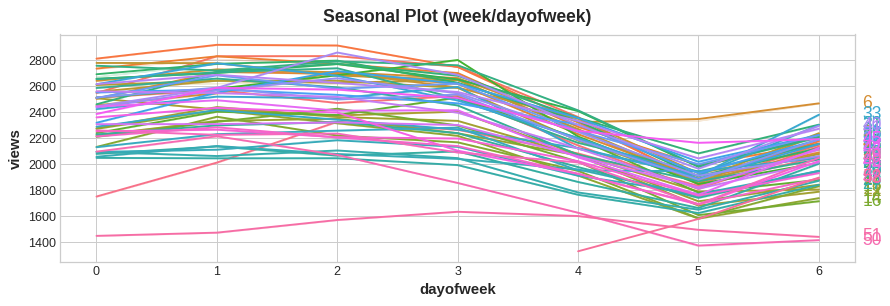
Dört zaman serisinde mevsimsel desenler.

Mevsimselliği modelleyen iki tür özellik öğreneceğiz. İlk tür olan **göstergeler (indicators)**, günlük gözlemlerden oluşan haftalık bir mevsim gibi az sayıda gözleme sahip mevsimler için en uygunudur. İkinci tür olan **Fourier özellikleri** ise, günlük gözlemlerden oluşan yıllık bir mevsim gibi çok sayıda gözleme sahip mevsimler için en uygunudur.

# Seasonal Plots and Seasonal Indicators[¶](https://www.kaggle.com/code/ryanholbrook/seasonality#Seasonal-Plots-and-Seasonal-Indicators)

Bir serideki trendi keşfetmek için hareketli ortalama grafiği kullandığımız gibi, mevsimsel kalıpları keşfetmek için de bir **mevsimsel grafik (seasonal plot)** kullanabiliriz.

Mevsimsel bir grafik, zaman serisinin bölümlerini ortak bir periyoda (gözlemlemek istediğiniz "mevsim") karşı gösterir. Şekil, Wikipedia'nın **Trigonometri** makalesinin günlük görüntülenmelerinin mevsimsel bir grafiğini göstermektedir: makalenin günlük görüntülenmeleri, ortak bir **haftalık** periyot boyunca çizilmiştir.



Bu seride belirgin bir haftalık mevsimsel model var; hafta içi yükseliyor ve hafta sonuna doğru düşüyor.

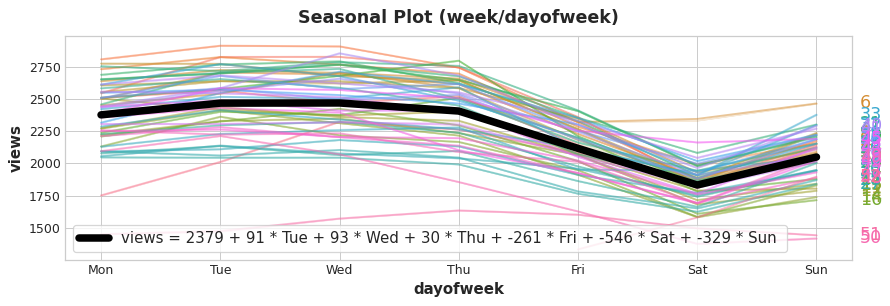
### **Seasonal indicators**[**¶**](https://www.kaggle.com/code/ryanholbrook/seasonality#Seasonal-indicators)

**Mevsimsel göstergeler**, bir zaman serisinin seviyesindeki mevsimsel farklılıkları temsil eden ikili (binary) özelliklerdir. Mevsimsel bir dönemi kategorik bir özellik gibi ele alıp tekil kodlama (one-hot encoding) uyguladığınızda elde ettiğiniz şey mevsimsel göstergelerdir.

Haftanın günlerini tekil kodlama ile kodlayarak, haftalık mevsimsel göstergeler elde ederiz. **Trigonometri** serisi için haftalık göstergeler oluşturmak bize altı yeni "kukla" özellik verecektir. (Doğrusal regresyon, göstergelerden birini atarsanız en iyi şekilde çalışır; aşağıdaki tabloda biz Pazartesi'yi seçtik.)

| Date | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday |
| --- | --- | --- | --- | --- | --- | --- |
| 2016-01-04 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2016-01-05 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2016-01-06 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2016-01-07 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 2016-01-08 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 2016-01-09 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| 2016-01-10 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 2016-01-11 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

Eğitim verilerine mevsimsel göstergelerin eklenmesi, modellerin mevsimsel bir dönem içindeki ortalamaları ayırt etmesine yardımcı olur:



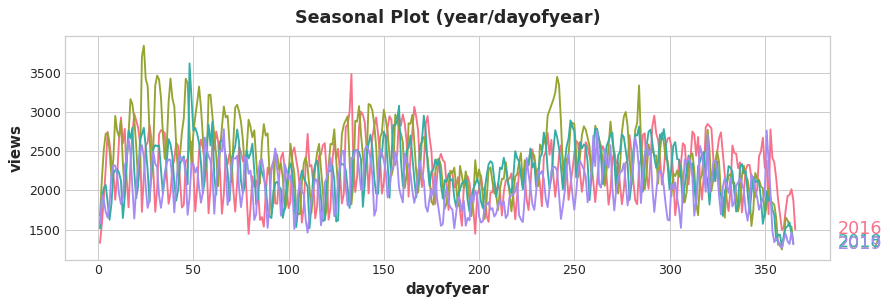
Sıradan doğrusal regresyon, mevsimin her anındaki ortalama değerleri öğrenir.

Göstergeler birer Açma/Kapama anahtarı gibi çalışır. Herhangi bir zamanda, bu göstergelerden en fazla biri **1** (Açık) değerine sahip olabilir. Doğrusal regresyon, **Mon** (Pazartesi) için **2379** gibi bir başlangıç değeri öğrenir ve ardından o gün için hangi gösterge **Açık** ise, onun değeri kadar ayarlama yapar; diğerleri **0** olur ve kaybolur.

# Fourier Features and the Periodogram[¶](https://www.kaggle.com/code/ryanholbrook/seasonality#Fourier-Features-and-the-Periodogram)

Şimdi ele alacağımız özellik türü, göstergelerin pratik olmayacağı, çok sayıda gözleme sahip uzun mevsimler için daha uygundur. **Fourier özellikleri**, her bir tarih için bir özellik oluşturmak yerine, mevsimsel eğrinin genel şeklini sadece birkaç özellikle yakalamaya çalışır.

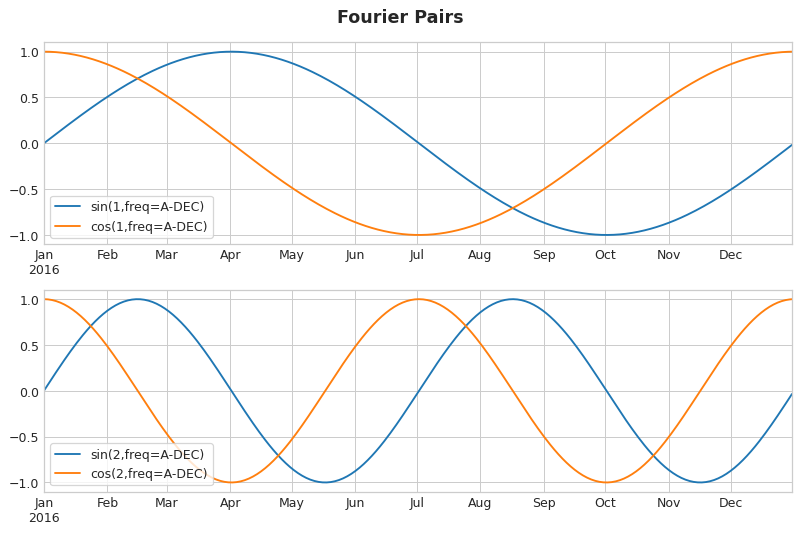
**Trigonometry**'deki yıllık mevsim için bir grafiğe bakalım. Çeşitli frekanslardaki tekrarları fark edin: yılda üç kez uzun bir yukarı-aşağı hareket, yılda 52 kez kısa haftalık hareketler ve belki de diğerleri.



Wiki Trigonometri serisinde yıllık mevsimsellik.

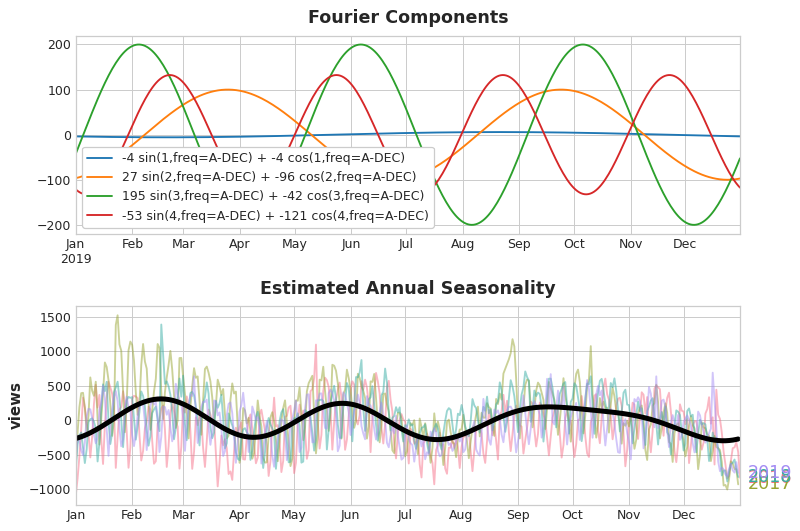
Bir mevsimin içindeki bu frekansları **Fourier özellikleri** ile yakalamaya çalışırız. Buradaki fikir, modellemeye çalıştığımız mevsimle aynı frekanslara sahip periyodik eğrileri eğitim verimize dahil etmektir. Kullandığımız eğriler, trigonometrik fonksiyonlardan sinüs ve kosinüs eğrileridir.

**Fourier özellikleri**, en uzun olanından başlayarak mevsimdeki her potansiyel frekans için bir çift sinüs ve kosinüs eğrisinden oluşur. Yıllık mevsimselliği modelleyen Fourier çiftleri şu frekanslara sahip olacaktır: yılda bir kez, yılda iki kez, yılda üç kez ve bu şekilde devam eder.



The first two Fourier pairs for annual seasonality. **Top:**Frequency of once per year. **Bottom:**Frequency of twice per year.

Eğer bu sinüs / kosinüs eğrilerinden oluşan bir seti eğitim verimize eklersek, doğrusal regresyon algoritması hedef serideki mevsimsel bileşeni yansıtacak ağırlıkları bulacaktır. Şekil, doğrusal regresyonun **Wiki Trigonometry** serisindeki yıllık mevsimselliği modellemek için dört Fourier çiftini nasıl kullandığını göstermektedir.

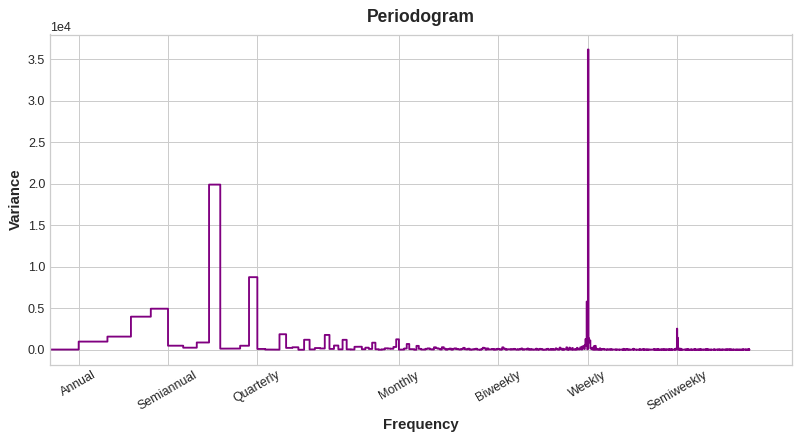


Üst: Regresyon katsayılı sinüs ve kosinüs toplamından oluşan dört Fourier çiftinin eğrileri. Her eğri farklı bir frekansı modellemektedir. Alt: Bu eğrilerin toplamı mevsimsel deseni yaklaşık olarak göstermektedir.

Yıllık mevsimselliğin iyi bir tahminini elde etmek için sadece sekiz özelliğe (dört sinüs / kosinüs çifti) ihtiyacımız olduğuna dikkat edin. Bunu, yüzlerce özellik (yılın her günü için bir tane) gerektirecek olan mevsimsel gösterge yöntemiyle karşılaştırın. Mevsimselliğin sadece "ana etkisini" Fourier özellikleri ile modelleyerek, eğitim verinize çok daha az özellik eklemeniz gerekir; bu da daha az hesaplama süresi ve aşırı uyum (overfitting) riskinin azalması anlamına gelir.

### **Choosing Fourier features with the Periodogram**[**¶**](https://www.kaggle.com/code/ryanholbrook/seasonality#Choosing-Fourier-features-with-the-Periodogram)

Özellik setimize tam olarak kaç tane Fourier çifti eklememiz gerektiğini periyodogram ile yanıtlayabiliriz. **Periyodogram**, bir zaman serisindeki frekansların gücünü gösterir. Daha açık olmak gerekirse, grafiğin y eksenindeki değer, (a2+b2)/2 formülüyle hesaplanır; burada a ve b, o frekanstaki sinüs ve kosinüsün katsayılarıdır (yukarıdaki "Fourier Bileşenleri" grafiğinde olduğu gibi).



*Periodogram for the*Wiki Trigonometry*series.*

Soldan sağa, periodogram çeyreklik dönemden sonra yılda dört kez düşüş gösteriyor. Bu nedenle yıllık sezonu modellemek için dört Fourier çifti seçtik. Haftalık sıklığı, göstergelerle daha iyi modellendiği için göz ardı ediyoruz.

### **Computing Fourier features (optional)**[**¶**](https://www.kaggle.com/code/ryanholbrook/seasonality#Computing-Fourier-features-(optional))

Fourier özelliklerinin nasıl hesaplandığını bilmek, onları kullanmak için zorunlu değildir, ancak detaylarını görmek konuyu daha iyi anlamanıza yardımcı olursa, aşağıdaki gizli hücre bir zaman serisinin indeksinden bir Fourier özellik setinin nasıl türetilebileceğini göstermektedir. (Ancak, uygulamalarımız için **statsmodels**'dan bir kütüphane fonksiyonu kullanacağız.)

import numpy as np

def fourier\_features(index, freq, order):

time = np.arange(len(index), dtype=np.float32)

k = 2 \* np.pi \* (1 / freq) \* time

features = {}

for i **in** range(1, order + 1):

features.update({

f"sin\_**{**freq**}**\_**{**i**}**": np.sin(i \* k),

f"cos\_**{**freq**}**\_**{**i**}**": np.cos(i \* k),

})

return pd.DataFrame(features, index=index)

*# Compute Fourier features to the 4th order (8 new features) for a*

*# series y with daily observations and annual seasonality:*

*#*

*# fourier\_features(y, freq=365.25, order=4)*

# Example - Tunnel Traffic[¶](https://www.kaggle.com/code/ryanholbrook/seasonality#Example---Tunnel-Traffic)

Tünel Trafiği veri setine bir kez daha devam edeceğiz. Bu gizli hücre verileri yükler ve iki işlevi tanımlar: seasonal\_plot ve plot\_periodogram.

from pathlib import Path

from warnings import simplefilter

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

from sklearn.linear\_model import LinearRegression

from statsmodels.tsa.deterministic import CalendarFourier, DeterministicProcess

simplefilter("ignore")

*# Set Matplotlib defaults*

plt.style.use("seaborn-whitegrid")

plt.rc("figure", autolayout=True, figsize=(11, 5))

plt.rc(

"axes",

labelweight="bold",

labelsize="large",

titleweight="bold",

titlesize=16,

titlepad=10,

)

plot\_params = dict(

color="0.75",

style=".-",

markeredgecolor="0.25",

markerfacecolor="0.25",

legend=False,

)

%config InlineBackend.figure\_format = 'retina'

*# annotations: https://stackoverflow.com/a/49238256/5769929*

def seasonal\_plot(X, y, period, freq, ax=None):

if ax **is** None:

\_, ax = plt.subplots()

palette = sns.color\_palette("husl", n\_colors=X[period].nunique(),)

ax = sns.lineplot(

x=freq,

y=y,

hue=period,

data=X,

ci=False,

ax=ax,

palette=palette,

legend=False,

)

ax.set\_title(f"Seasonal Plot (**{**period**}**/**{**freq**}**)")

for line, name **in** zip(ax.lines, X[period].unique()):

y\_ = line.get\_ydata()[-1]

ax.annotate(

name,

xy=(1, y\_),

xytext=(6, 0),

color=line.get\_color(),

xycoords=ax.get\_yaxis\_transform(),

textcoords="offset points",

size=14,

va="center",

)

return ax

def plot\_periodogram(ts, detrend='linear', ax=None):

from scipy.signal import periodogram

fs = pd.Timedelta("365D") / pd.Timedelta("1D")

freqencies, spectrum = periodogram(

ts,

fs=fs,

detrend=detrend,

window="boxcar",

scaling='spectrum',

)

if ax **is** None:

\_, ax = plt.subplots()

ax.step(freqencies, spectrum, color="purple")

ax.set\_xscale("log")

ax.set\_xticks([1, 2, 4, 6, 12, 26, 52, 104])

ax.set\_xticklabels(

[

"Annual (1)",

"Semiannual (2)",

"Quarterly (4)",

"Bimonthly (6)",

"Monthly (12)",

"Biweekly (26)",

"Weekly (52)",

"Semiweekly (104)",

],

rotation=30,

)

ax.ticklabel\_format(axis="y", style="sci", scilimits=(0, 0))

ax.set\_ylabel("Variance")

ax.set\_title("Periodogram")

return ax

data\_dir = Path("../input/ts-course-data")

tunnel = pd.read\_csv(data\_dir / "tunnel.csv", parse\_dates=["Day"])

tunnel = tunnel.set\_index("Day").to\_period("D")

Bir hafta ve bir yıl boyunca mevsimsel grafiklere bir göz atalım.

X = tunnel.copy()

*# days within a week*

X["day"] = X.index.dayofweek *# the x-axis (freq)*

X["week"] = X.index.week *# the seasonal period (period)*

*# days within a year*

X["dayofyear"] = X.index.dayofyear

X["year"] = X.index.year

fig, (ax0, ax1) = plt.subplots(2, 1, figsize=(11, 6))

seasonal\_plot(X, y="NumVehicles", period="week", freq="day", ax=ax0)

seasonal\_plot(X, y="NumVehicles", period="year", freq="dayofyear", ax=ax1);

Şimdi periodograma bakalım:

plot\_periodogram(tunnel.NumVehicles);

Periodogram yukarıdaki mevsimsel grafiklerle uyuşuyor: güçlü bir haftalık mevsim ve daha zayıf bir yıllık mevsim. Haftalık mevsimi göstergelerle, yıllık mevsimi ise Fourier özellikleri ile modelleyeceğiz. Sağdan sola doğru, periyodogram **İki Aylık (6)** ve **Aylık (12)** arasında düşüşe geçiyor, bu yüzden 10 Fourier çifti kullanalım.

Mevsimsel özelliklerimizi, 2. Derste trend özelliklerini oluşturmak için kullandığımız yardımcı program olan **DeterministicProcess**'i kullanarak oluşturacağız. İki mevsimsel periyodu (haftalık ve yıllık) kullanmak için, bunlardan birini "ek bir terim" olarak örneklememiz (instantiate) gerekecek:

from statsmodels.tsa.deterministic import CalendarFourier, DeterministicProcess

fourier = CalendarFourier(freq="A", order=10) *# 10 sin/cos pairs for "A"nnual seasonality*

dp = DeterministicProcess(

index=tunnel.index,

constant=True, *# dummy feature for bias (y-intercept)*

order=1, *# trend (order 1 means linear)*

seasonal=True, *# weekly seasonality (indicators)*

additional\_terms=[fourier], *# annual seasonality (fourier)*

drop=True, *# drop terms to avoid collinearity*

)

X = dp.in\_sample() *# create features for dates in tunnel.index*

Özellik setimiz oluşturulduktan sonra, modeli uygulamaya ve tahminler yapmaya hazırız. Modelimizin eğitim verilerinin ötesine nasıl geçtiğini görmek için 90 günlük bir tahmin ekleyeceğiz. Buradaki kod, önceki derslerdekiyle aynı.

y = tunnel["NumVehicles"]

model = LinearRegression(fit\_intercept=False)

\_ = model.fit(X, y)

y\_pred = pd.Series(model.predict(X), index=y.index)

X\_fore = dp.out\_of\_sample(steps=90)

y\_fore = pd.Series(model.predict(X\_fore), index=X\_fore.index)

ax = y.plot(color='0.25', style='.', title="Tunnel Traffic - Seasonal Forecast")

ax = y\_pred.plot(ax=ax, label="Seasonal")

ax = y\_fore.plot(ax=ax, label="Seasonal Forecast", color='C3')

\_ = ax.legend()

Tahminlerimizi iyileştirmek için zaman serileriyle yapabileceğimiz daha çok şey var. Bir sonraki derste, zaman serilerini birer özellik olarak nasıl kullanacağımızı öğreneceğiz. Zaman serilerini bir tahminin girdisi olarak kullanmak, serilerde sıklıkla bulunan bir diğer bileşeni, yani döngüleri modellememizi sağlar.