T.C. YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Proje

Hazırlayan : ALİ RÜVEYCAN

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Haberleşme Tezli Yüksek Lisans Programı

Veri Güdümlü Vekil Modelleme Teknikleri ve Uygulamaları

Proje Konusu : 5G için Yanlarında SRR hücreleri bulunan mikroşerit antenin vekil tabanlı tasarımı

Proje kapsamında, metamalzemeli mikroşerit antenin elektromanyetik simülasyonlarına dayanan veri üretimi aşaması ve vekil modelleme aşaması gösterilmiştir.
Çalışmada Square SRR kullanılmıştır.

1.CST'nin Başlatılması

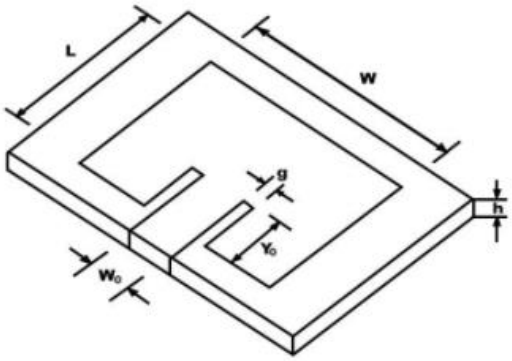
```
addpath(genpath('CST-MATLAB-API-master'));

cst = actxserver('CSTStudio.application');

mws = cst.invoke('NewMWS');

fcenter = 3.5;    %merkez frekans
fmin = fcenter * 0.97

fmax = fcenter * 1.03

CstDefineUnits(mws, 'um', 'GHz', 's', 'Kelvin', 'V', 'A', 'Ohm', 'S', 'PikoF',
'NanoH');
CstDefineFrequencyRange(mws, fmin, fmax);
CstMeshInitiator(mws);

%Sınır Koşulları
Xmin = 'expanded open';
Xmax = 'expanded open';
Ymin = 'expanded open';
Ymax = 'expanded open';
Zmin = 'expanded open';
Zmax = 'expanded open';


CstDefineOpenBoundary(mws, fmin, Xmin, Xmax, Ymin, Ymax, Zmin, Zmax);

%arka plan malzemesini tanımlıyoruz


XminSpace = 0;
XmaxSpace = 0;
YminSpace = 0;
YmaxSpace = 0;
ZminSpace = 0;
ZmaxSpace = 0;


CstDefineBackroundMaterial(mws, XminSpace, XmaxSpace, YminSpace, YmaxSpace,
ZminSpace, ZmaxSpace);
```

2.Malzemelerin Tanımı

```
metal = CstCopperAnnealedLossy(mws);

[dieletric, epsilon_r] = CstFR4lossy(mws);

t = 0.035*1e3; %metal levhanın yüksekliği
```

3.Temel Antenin Tasarımı

```
h = 1700;   %alt tabakanın yüksekliği
```



Mikroşerit antenin geometrisi ile ilişkili değişkenler

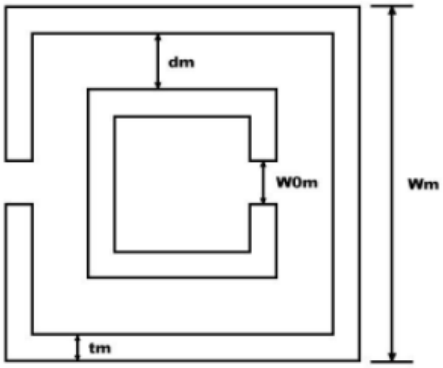
```
W = 0.9*3e8/(2*fcenter*1e9)*sqrt(2/(epsilon_r+1))*1e6; %radyasyon verimliliği

epsilon_reff = (epsilon_r+1)/2+((epsilon_r-1)/2).*(1+(12/W).*h).^(-1/2);
%antenin etkin dielektrik sabiti

delta_l = 0.412.*h.*((epsilon_reff+0.3).*(W./h+0.264))./((epsilon_reff-
0.258).*(W./h+0.8));
%yamanın gerçekte olduğundan ne kadar büyük göründüğü

l = 0.98.*3e5./(2*fcenter*sqrt(epsilon_reff))-2.*delta_l;

Y0 = 4*h;
g = 1000;
W0 = 3.5*g;

DrawMicrostripAntenna(mws, W, l, t, h, Y0, W0, g, metal, dieletric);
DrawPort(mws, W, l, t, h, W0);
```

4.Temel Antenin Simülasyonu

```
base_path = 'C:\Kullanıcılar\aliru\results';
 %sonuçların dışa aktarılacağı yol
CstDefineFarfieldMonitor(mws,strcat('farfield (f=',num2str(fcenter),')'),
fcenter);

%Uzak alan monitörünü rezonans frekansında ayarlamak için CST'de Post-
Processing>Result Templates>Template Based Post Processing seçiyoruz. İlk
kutuda farfield ve anten özelliklerini, ikinci kutuda farfield sonucunu
seçiyoruz. Frequency üzerinden Maksimum Kazanç Şablonunu seçerek onu "Kazanç"
olarak adlandırıyoruz.
```

```
CstDefineTimedomainSolver(mws,-40);
ExportResults(mws, base_path, 0, W, l, t, h, Y0, W0, g);
CstSaveProject(mws,strcat(base_path, '\0\simula.cst'));
CstQuitProject(mws);
```

5.Metamalzeme Hücresi İçin Değer Aralığı



Kare metamalzemesi hücresinin geometrisiyle ilişki değişkenler

Metamalzeme hücreleri DNG(çift negatif) özelliklerini şu aralıklar için sergiler :

$$0,025\lambda \leq W_m \leq \frac{\lambda}{4}$$

$$t_m \approx 0,1 W_m$$

$$1,9 \times 10^{-3}\lambda \leq W_{0m} \leq 7,6 \times 10^{-3}\lambda$$

$$0,9 \times 10^{-3}\lambda \leq d_m \leq 5,7 \times 10^{-3}\lambda$$

lambda = c/f

```
lambda = 3e5./fcenter;
n_Wm = 5;

Wm_min = 0.025*lambda;
Wm_max = 1/4*lambda;
Wm_step = (Wm_max-Wm_min)/(n_Wm-1);
Wm = Wm_min:Wm_step:Wm_max;

tm = 0.1.*Wm

n_W0m = 4;
W0m_min = 1.9e-3*lambda;
W0m_max = 7.6e-3*lambda;
W0m_step = (W0m_max-W0m_min)/(n_W0m-1);
W0m = W0m_min:W0m_step:W0m_max;

n_dm = 4;
dm_min = 0.9e-3*lambda;
dm_max = 5.7e-3*lambda;
dm_step = (dm_max-dm_min)/(n_dm-1);
dm = dm_min:dm_step:dm_max;

n_sim = n_Wm*n_W0m*n_dm;
```

6.Pro-Start Değerlerinin Aralığı

```
rows = 3:2:7;
```

$$0 \le X_a \le \left(\frac{W}{2} - \frac{W_m}{2}\right)$$

```
&Xa yaması arasındaki X cinsinden mesafe

n_Xa = 2;
Xa = zeros(length(Wm), n_Xa);
for i = 1:length(Wm)
    Xa_min = 0;
    Xa_max = W/2-Wm(i)/2;
    Xa_step = (Xa_max-Xa_min)/(n_Xa-1);

    Xa(i,:) = Xa_min:Xa_step:Xa_max;
end
Xa = Flat(Xa)
```

$$W_m \le Y_a \le \left(\frac{4L}{\text{rows} - 1} - W_m\right)$$

```
%hücreler arasındaki Ya mesafesi

n_Ya = 2;
Ya = zeros(length(Wm), n_Ya);
for i = 1:length(Wm)
    Ya_min = Wm(i);
    Ya_max = 4*l/(min(rows) - 1)-Wm(i);
    Ya_step = (Ya_max-Ya_min)/(n_Ya-1);

    Ya(i,:) = Ya_min:Ya_step:Ya_max;
end
Ya = Flat(Ya);
Ya = Ya(Ya>=0)
```

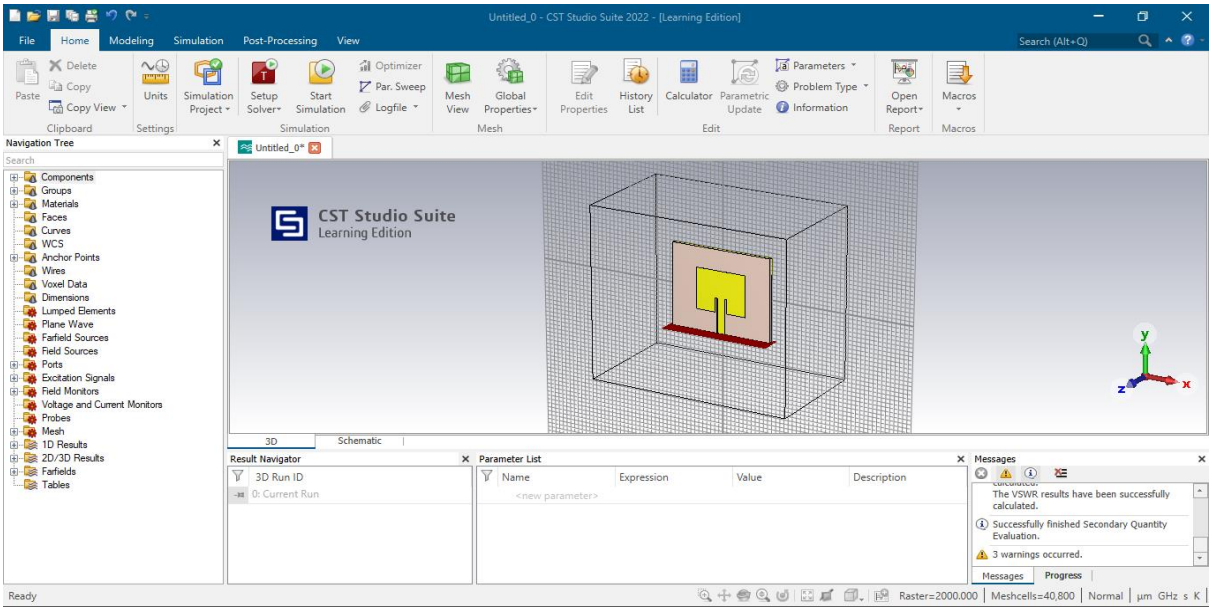
7.Simülasyon

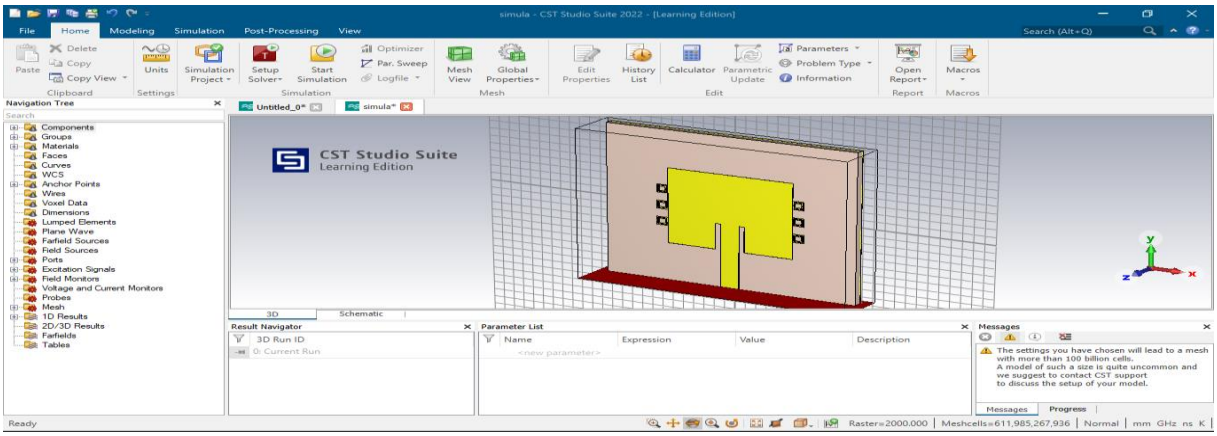
Hangi geometri kombinasyonlarının mümkün olduğunu kontrol etmek için 1:1 yaparız.

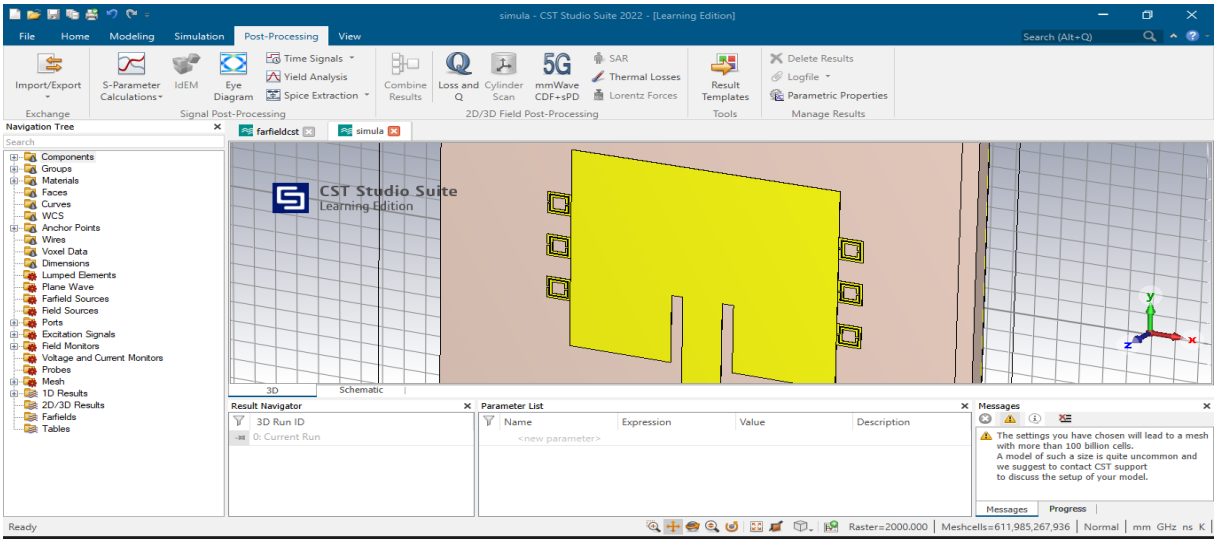
```
possible_geometry = [];
count = 1;
for wm = 1:length(Wm)
    for w0m = 1:length(W0m)
        for Dm = 1:length(dm)

            for row = 1:length(rows)
                for xa = 1:length(Xa)
                    if (xa <= W/2 - Wm(wm)/2) % dizilimin alt tabaka
genişliğinin ötesine geçmediği anlamına gelir.
                        for ya = 1:length(Ya)
                            if(ya <= 4*l/(rows(row)-1) - Wm(wm)) % dizilimin
alt tabaka yüksekliğinin ötesine geçmediği anlamına gelir.
                                possible_geometry(count, :) = [Wm(wm) tm(wm)
W0m(w0m) dm(Dm) rows(row) Xa(xa) Ya(ya)];
```

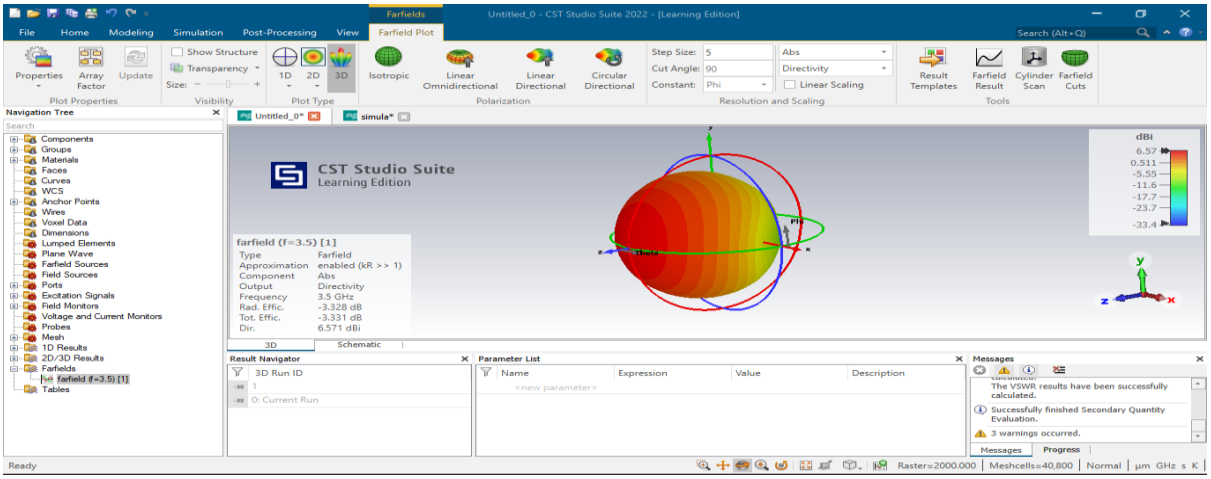
```
                                count = count + 1;
                            end
                        end
                    end
                end
            end
        end
    end
end


fprintf(['Simulasyon Sayisi: %d\n' ...
    'Tahmini tamamlanma suresi: %.0f dias'], count, 3*count/60/24);

%olası düzenlemeleri oluşturmak için şunları yapmalıyız:

count = 1;
for i = 1:length(possible_geometry)

    mws = cst.invoke('OpenFile', strcat(base_path, '\0\simula.cst'));

    CstDefineFrequencyRange(mws, fmin, fmax);
    CstMeshInitiator(mws);

    CstDefineOpenBoundary(mws, fmin, Xmin, Xmax, Ymin, Ymax, Zmin, Zmax);

    CstDefineBackroundMaterial(mws, XminSpace, XmaxSpace, YminSpace,
YmaxSpace, ZminSpace, ZmaxSpace);

    metal = CstCopperAnnealedLossy(mws);
    [dieletric, epsilon_r] = CstFR4lossy(mws);

    DrawPort(mws, W, l, t, h, W0);

    for side = [-1 1]
        for cell = 1:possible_geometry(i,5)

            Wm = possible_geometry(i,1);
            W0m = possible_geometry(i,3);
            dm = possible_geometry(i,4);
            tm = possible_geometry(i,2);
            hm = t+h;
            h0 = t;

            Xa = possible_geometry(i,6);
            Ya = possible_geometry(i,7);

            if cell == 1
                ya = 0;
            elseif mod(cell,2) == 0
                ya = (cell/2)*(Ya+Wm);
            else
```

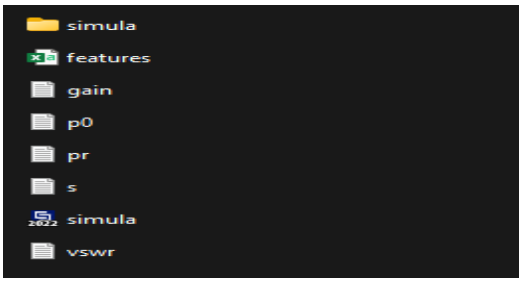
```
                ya = (-1)*((cell-1)/2)*(Ya+Wm);
            end

            xa = (W/2+Xa+Wm/2)*side;

            DrawSquareSRR(mws,Wm,W0m,dm,tm,hm,h0,metal,cell*side,xa,ya)
        end
    end

    CstDefineFarfieldMonitor(mws,strcat('farfield (f=',num2str(fcenter),')'),
fcenter);

    CstDefineTimedomainSolver(mws,-40);

    ExportResultsMeta(mws, 'C:\Kullanıcılar\aliru\results', count,
Wm,W0m,dm,tm,hm,h0,cell,Xa,Ya);


CstSaveAsProject(mws,strcat('C:\Kullanıcılar\aliru\results\',num2str(count),'\
simula.cst'));
    CstQuitProject(mws);

    count = count + 1;
end
```




simula - CST Studio Suite 2022 - [Learning Edition]
CST Studio Suite
Learning Edition


simula - CST Studio Suite 2022 - [Learning Edition]
CST Studio Suite
Learning Edition


Untitled_0 - CST Studio Suite 2022 - [Learning Edition]
CST Studio Suite
Learning Edition
farfield (f=3.5) [1]

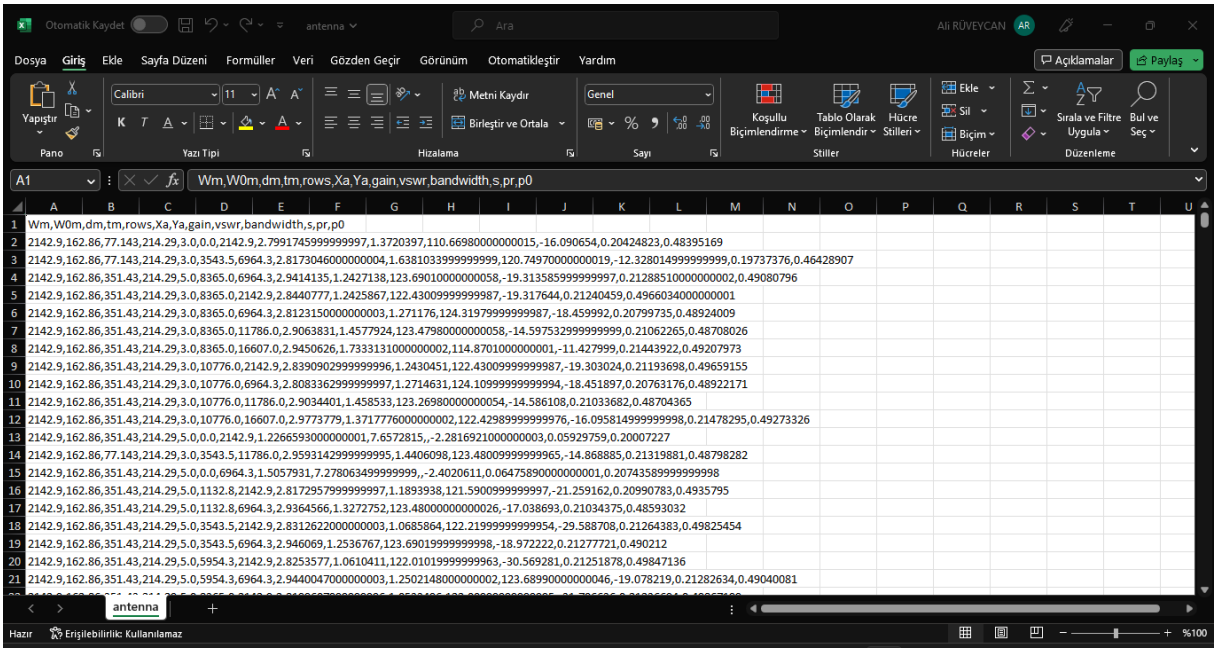

simula
features
gain
p0
pr
s
simula
vswr

8. Veri Seti Oluşturma

Python dilinde PyCharm derleyicisi kullanarak aşağıdaki kodlar ile antenna.csv data seti oluşturuldu.

```
import pyprind
import pandas as pd
import numpy as np
import os

base_path = 'C:/Kullanıcılar/aliru/results/0'

results = np.array(os.listdir(base_path))
results = np.delete(results, [0], axis=0)

n_simulations = results.astype(int).max()

pbar = pyprind.ProgBar(n_simulations)

df_features = pd.DataFrame()
df_targets = pd.DataFrame()

files = ['features.csv', 'p0.txt', 'pr.txt', 's.txt',
'vswr.txt', 'gain.txt']

for simulation in results:
    df_targets_simulation = pd.DataFrame()

    for file in files:
        with open(base_path+ '/{}/{}'.format(simulation,
file), 'r') as infile:
            # for csv's
            if infile.name[-4:] == '.csv':
                df_features =
df_features.append([infile.read().split(',')],
ignore_index=True)
                df_features = df_features.astype('float64')

            # for txt's
            if infile.name[-4:] == '.txt':
                df_txt = pd.read_csv(infile.name,
skiprows=[0,1], header=None, delim_whitespace=True)

                value = file.split('.')[0]

                df_txt.columns = ['f', value]

                row_f = df_txt[df_txt.f == 3.5]
                row_f.reset_index(inplace=True)
```

```
                df_targets_simulation.insert(loc=0,
column=value, value=row_f[value])

                # getting bandwidth
                if value == 's':
                    df_util = df_txt.loc[df_txt['s'] <= (-10)]

                    bandwidth = (df_util.f.max() -
df_util.f.min())*1000

                    df_targets_simulation.insert(loc=0,
column='bandwidth', value=bandwidth)

            pbar.update()

    df_targets = df_targets.append(df_targets_simulation,
ignore_index=True)


df_features.columns = ['Wm', 'W0m', 'dm', 'tm', 'hm', 'h0',
'rows', 'Xa', 'Ya']

df = df_features.join(df_targets)

df = df.drop(['hm', 'h0'], axis=1)

print(df.head(),'\n',df.info(), '\n', df.describe())

df.to_csv('./antenna.csv', index=False)
```



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Wm,W0m,dm,tm,rows,Xa,Ya,gain,vswr,bandwidth,s,pr,p0																				
2	2142.9,162.86,77.143,214.29,3.0,0.0,2142.9,2.7991745999999997,1.3720397,110.66980000000015,-16.090654,0.20424823,0.48395169																				
3	2142.9,162.86,77.143,214.29,3.0,3543.5,6964.3,2.8173046000000004,1.6381033999999999,120.74970000000019,-12.328014999999999,0.19737376,0.46428907																				
4	2142.9,162.86,351.43,214.29,5.0,8365.0,6964.3,2.9414135,1.2427138,123.69010000000058,-19.313585999999997,0.21288510000000002,0.49080796																				
5	2142.9,162.86,351.43,214.29,3.0,8365.0,2142.9,2.8440777,1.2425867,122.43009999999987,-19.317644,0.21240459,0.4966034000000001																				
6	2142.9,162.86,351.43,214.29,3.0,8365.0,6964.3,2.8123150000000003,1.271176,124.31979999999987,-18.459992,0.20799735,0.48924009																				
7	2142.9,162.86,351.43,214.29,3.0,8365.0,11786.0,2.9063831,1.4577924,123.47980000000058,-14.597532999999999,0.21062265,0.48708026																				
8	2142.9,162.86,351.43,214.29,3.0,8365.0,16607.0,2.9450626,1.7333131000000002,114.8701000000001,-11.427999,0.21443922,0.49207973																				
9	2142.9,162.86,351.43,214.29,3.0,10776.0,2142.9,2.8390902999999996,1.2430451,122.43009999999987,-19.303024,0.21193698,0.49659155																				
10	2142.9,162.86,351.43,214.29,3.0,10776.0,6964.3,2.8083362999999997,1.2714631,124.10999999999994,-18.451897,0.20763176,0.48922171																				
11	2142.9,162.86,351.43,214.29,3.0,10776.0,11786.0,2.9034401,1.458533,123.26980000000054,-14.586108,0.21033682,0.48704365																				
12	2142.9,162.86,351.43,214.29,3.0,10776.0,16607.0,2.9773779,1.3717776000000002,122.42989999999976,-16.095814999999998,0.21478295,0.49273326																				
13	2142.9,162.86,351.43,214.29,5.0,0.0,2142.9,1.2266593000000001,7.6572815,,-2.2816921000000003,0.05929759,0.20007227																				
14	2142.9,162.86,77.143,214.29,3.0,3543.5,11786.0,2.9593142999999995,1.4406098,123.48009999999965,-14.868885,0.21319881,0.48798282																				
15	2142.9,162.86,351.43,214.29,5.0,0.0,6964.3,1.5057931,7.278063499999999,,-2.4020611,0.06475890000000001,0.20743589999999998																				
16	2142.9,162.86,351.43,214.29,5.0,1132.8,2142.9,2.8172957999999997,1.1893938,121.5900999999997,-21.259162,0.20990783,0.4935795																				
17	2142.9,162.86,351.43,214.29,5.0,1132.8,6964.3,2.9364566,1.3272752,123.48000000000026,-17.038693,0.21034375,0.48593032																				
18	2142.9,162.86,351.43,214.29,5.0,3543.5,2142.9,2.8312622000000003,1.0685864,122.21999999999954,-29.588708,0.21264383,0.49825454																				
19	2142.9,162.86,351.43,214.29,5.0,3543.5,6964.3,2.946069,1.2536767,123.69019999999998,-18.972222,0.21277721,0.490212																				
20	2142.9,162.86,351.43,214.29,5.0,5954.3,2142.9,2.8253577,1.0610411,122.01019999999963,-30.569281,0.21251878,0.49847136																				
21	2142.9,162.86,351.43,214.29,5.0,5954.3,6964.3,2.9440047000000003,1.2502148000000002,123.68990000000046,-19.078219,0.21282634,0.49040081																				

antenna

Hazır Erişilebilirlik: Kullanılamaz %100

9. Vekil Modelleme

GİRİŞ

Elektromanyetik simülatörlere bir alternatif oluşturacağız. Bu şekilde, girdimiz antenin geometrik boyutları olacak, tahminimiz ise antenin bazı özellikleri hakkında olacak. Bu durumda, bir regresyon problemimiz var.

Bağımsız Değişkenler

Birim hücre değişkenleri hakkında :

Wm : Kare SRR hücre yüksekliği (mm cinsinden)
W0m : Halka boşluğu (mm cinsinden)
dm : Halkalar arası mesafe (mm cinsinden)
tm : Halka kalınlığı (mm cinsinden)

Düzenlemenin Değişkenleri Hakkında

Xa : Düzenlemeden yamaya olan X cinsinden mesafe (mm cinsinden)
Ya : Düzenlemenin hücreleri arasındaki Y cinsinden mesafe (mm cinsinden)
rows : Düzenlemedeki hücre sayısı (her iki tarafta)

Bağımlı Değişkenler

gain : Anten Kazancı (dB cinsinden)
bandwidth : Bant Genişliği (MHz cinsinden)
s : Geri Dönüş Kaybı (dB cinsinden)

Verileri ve Kütüphaneleri İçe Aktarma

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas import Index
from sklearn.model_selection import train_test_split,
GridSearchCV
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler, Normalizer
from sklearn.linear_model import Lasso
from sklearn.neighbors import KNeighborsRegressor
from sklearn.pipeline import Pipeline
import warnings
warnings.filterwarnings('ignore')


data = pd.read_csv('antenna.csv')
print(data.head())
```

Çıktı:

```
      Wm     W0m       dm      tm  ...  bandwidth          s        pr        p0
0  2142.9  162.86   77.143  214.29  ...   110.6698 -16.090654  0.204248  0.483952
1  2142.9  162.86   77.143  214.29  ...   120.7497 -12.328015  0.197374  0.464289
2  2142.9  162.86  351.430  214.29  ...   123.6901 -19.313586  0.212885  0.490808
3  2142.9  162.86  351.430  214.29  ...   122.4301 -19.317644  0.212405  0.496603
4  2142.9  162.86  351.430  214.29  ...   124.3198 -18.459992  0.207997  0.489240

[5 rows x 13 columns]
```

a) Veri Temizleme ve Hazırlama

```
data.columns
Index(['Wm', 'W0m', 'dm', 'tm', 'rows', 'Xa', 'Ya', 'gain',
'vswr',
'bandwidth', 's', 'pr', 'p0'],
dtype='object')
print(data.shape)
```

Çıktı:

```
(572, 13)
```

```
print(data.describe())
```

Çıktı:

```
                Wm         W0m          dm  ...           s          pr          p0
count   572.000000  572.000000  572.000000  ...  572.000000  572.000000  572.000000
mean   2244.048252  400.594178  275.425776  ...  -16.104948    0.192872    0.456955
std     691.578895  184.905214  150.901130  ...    7.897142    0.046994    0.089213
min    2142.900000  162.860000   77.143000  ...  -33.903172    0.037383    0.186297
25%    2142.900000  162.860000   77.143000  ...  -21.321550    0.200121    0.470947
50%    2142.900000  325.710000  214.290000  ...  -14.910812    0.211927    0.490360
75%    2142.900000  488.570000  351.430000  ...  -11.498286    0.214335    0.497100
max    6964.300000  651.430000  488.570000  ...   -2.083432    0.229538    0.499826

[8 rows x 13 columns]
```

```
print(data.info())
```

Çıktı:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 572 entries, 0 to 571
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Wm          572 non-null    float64
 1   W0m         572 non-null    float64
 2   dm          572 non-null    float64
 3   tm          572 non-null    float64
 4   rows        572 non-null    float64
 5   Xa          572 non-null    float64
 6   Ya          572 non-null    float64
 7   gain        572 non-null    float64
 8   vswr        572 non-null    float64
 9   bandwidth   509 non-null    float64
 10  s           572 non-null    float64
 11  pr          572 non-null    float64
 12  p0          572 non-null    float64
dtypes: float64(13)
memory usage: 58.2 KB
```

Bant genişliği olmayan simülasyonlar, kullanılan antenin bant genişliğine sahip olmadığı anlamına gelir. Kabul edilen frekansta rezonansa girer. (Çalışmaz)
Öyleyse, boş değerleri temizleyelim.

Boş Değerleri Temizleme

Burada, bant genişliği olmayan her satırı atıyoruz.

```
data = data.dropna()
print(data.shape)
```

Çıktı:

```
(509, 13)
```

b) Veri Ayrıştırma

Özellikler (X) ve Hedefler (Y) sütunlarını ayırıyoruz.

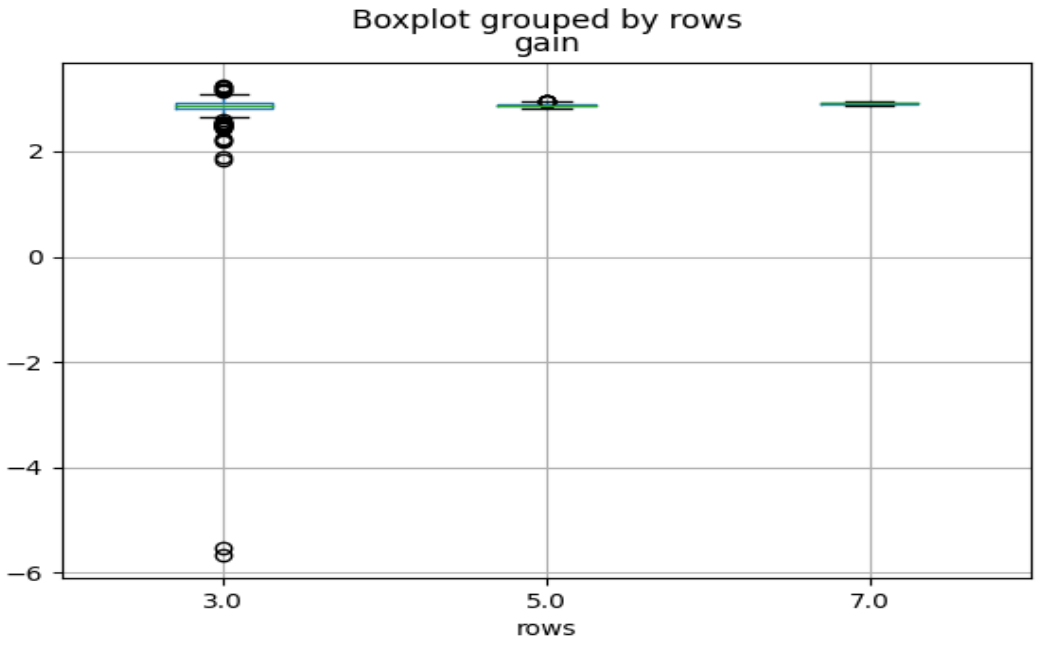
```
X_columns = ['Wm', 'W0m', 'dm', 'tm', 'Xa', 'Ya', 'rows']
y_columns = ['gain', 'bandwidth', 's']
X = data[X_columns]
y = data[y_columns]
```

c) Kazanç İçin Bir Model Oluşturmak

Kazancı Y'den ayıralım.

```
gain = y['gain']
data.boxplot('gain', 'rows')
plt.show()
```

Çıktı:



Burada 3 satır olduğunda bazı aykırı değerler görebiliriz. Bu tür bilgiler kullanışlı değildir ve genellikle modellemeyi bozar.

gain > 0 koşulunu sağlamayan satırları kaldıralım.

```
X = X[gain > 0]
gain = gain[gain > 0]
```

Özelliklerin varyansını kontrol ettiğimizde şunu elde ederiz :

```
print(X.var())
```

Çıktı:

```
Wm      3.171436e+05
W0m     3.427141e+04
dm      2.280619e+04
tm      3.171436e+03
Xa      9.805929e+06
Ya      2.821793e+07
rows    1.998020e+00
dtype: float64
```

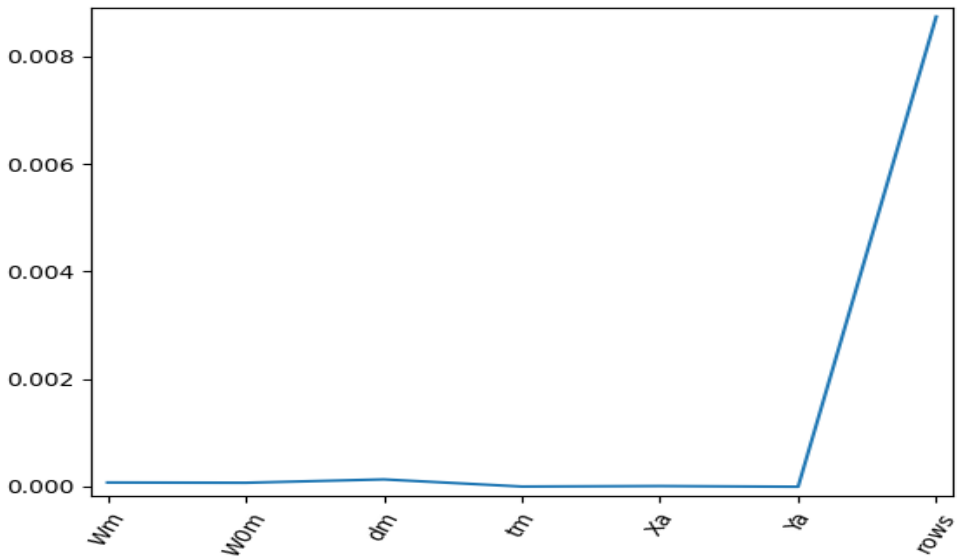
Satırlar hariç tüm özelliklerin varyansının oldukça yüksek olduğunu görebilirsiniz. Bu, modellemeden önce verileri normalleştirmemiz gerektiğinin bir işaretidir.

d) Özellik Seçimi

Lasso Regresyon algoritmaları aracılığıyla hangi özelliğin kazanç üzerinde daha fazla etkiye sahip olduğunu kontrol ediyoruz:

```
#Lasso Regresyonu üzerinde hiperparametre ayarı gerçekleştiriyoruz.
param_grid = {'alpha': np.arange(1e-4, 1e-3, 1e-4)}
lasso = Lasso(True)
lasso_cv = GridSearchCV(lasso, param_grid, cv=5)
lasso_cv.fit(X, gain)

#En iyi alpha'yı ayarlıyoruz.
print('Kazanç Lasso için en iyi alpha {}
dır'.format(lasso_cv.best_params_['alpha']))
lasso.alpha = lasso_cv.best_params_['alpha']
lasso.fit(X, gain)

Çıktı:
 Kazanç Lasso için en iyi alpha 0.0009000000000000001 dır

#Lasso'dan katsayıları alıyoruz.
lasso_coef = lasso.coef_
plt.plot(range(len(X_columns)), lasso_coef)
plt.xticks(range(len(X_columns)), X_columns, rotation=60)
plt.margins(0.02)
plt.show()
```

Çıktı:



Özellik satırlarının kazanç üzerinde en fazla etkiye sahip olduğu görülüyor. Bu iki değişkenin doğru orantılı bir ilişkiye sahip olduğunu görebilirsiniz. Wm ile minimum ilişkiyi de gözlemlemek mümkündür. Kazanç için 2 durum vardır:
1- Düzenlemede ne kadar çok SRR hücresi varsa, kazanç o kadar yüksek olur.
2-Kare SRR hücresinin yüksekliği (Wm) ne kadar düşükse, kazanç o kadar yüksek olur.

Model Eğitimi

Aşağıdaki adımlarla bir pipeline oluşturacağız:

Scaler ------ Verilerin ölçeğini değiştirmek için
KNeighborsRegressor ---------- Bir regresyon algoritması olarak

```
X_train, X_test, y_train, y_test = train_test_split(X, s,
random_state=42)
```

Pipeline için adımlar

```
steps = [('normalizer', Normalizer()),
('scaler', StandardScaler()),
('knn', KNeighborsRegressor())]
pipeline = Pipeline(steps)
```

Çapraz doğrulama için parametreler

```
param_grid = {'knn__n_neighbors': np.arange(1, 8)}
cv = GridSearchCV(pipeline, param_grid)
cv.fit(X_train, y_train)
y_pred = cv.predict(X_test)
print('En iyi parametreler: {}'.format(cv.best_params_))
print('R^2: {}'.format(cv.score(X_test, y_test)))
print('np.mean(R^2): {}'.format(mean_squared_error(y_test,
y_pred)))
```

Çıktı:

```
En iyi parametreler: {'knn__n_neighbors': 1}
R^2: 0.7708413831601835
np.mean(R^2): 0.0036264643897079 51
```

```
En iyi modeli kaydetmek :
gain_best = cv.best_estimator_
```

e) Geri Dönüş Kaybı için Bir Model Oluşturma

s'yi y'den ayıralım.

```
X = data[X_columns]
s = y['s']
print(s.shape, X.shape)
```

Çıktı:

```
(509,) (509, 7)
```

Lasso regresyon algoritması ile hangi özelliğin geri dönüş kaybı üzerinde daha fazla etkiye sahip olduğunu kontrol ediyoruz.

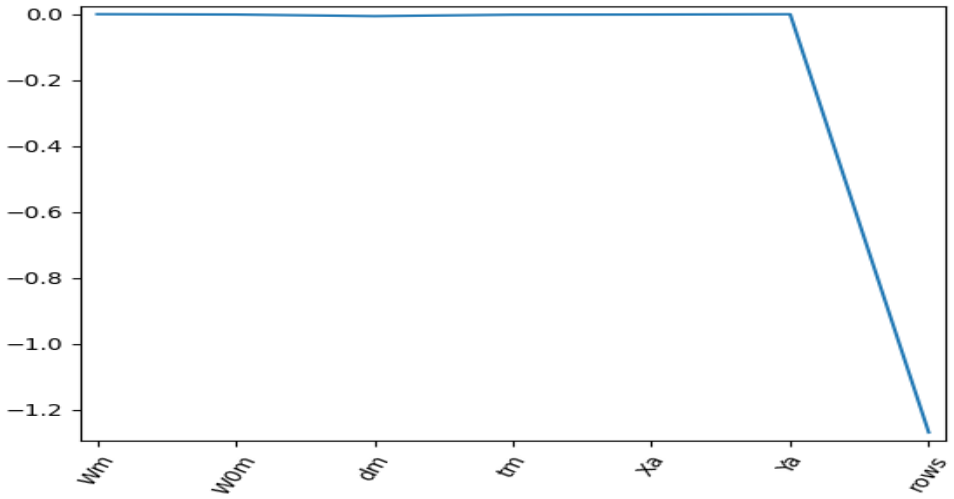
```
#Lasso Regresyonu üzerinde hiperparametre ayarı gerçekleştiriyoruz.
param_grid = {'alpha': np.arange(1e-4, 1e-3, 1e-4)}
lasso = Lasso(True)
lasso_cv = GridSearchCV(lasso, param_grid, cv=5)
lasso_cv.fit(X, s)

#En iyi alpha'yı ayarlıyoruz.
print('geri donus kaybi Lasso için en iyi alpha {}
dır'.format(lasso_cv.best_params_['alpha']))
lasso.alpha = lasso_cv.best_params_['alpha']
lasso.fit(X, s)
```

Çıktı:

geri donus kaybi Lasso için en iyi alpha 0.0001 dır.

```
lasso_coef = lasso.coef_
plt.plot(range(len(X_columns)), lasso_coef)
plt.xticks(range(len(X_columns)), X_columns, rotation=60)
plt.margins(0.02)
plt.show()
```

Çıktı:



Model Eğitimi

Tahmin modeli olarak yine KneighborsRegressor kullandığımızda, şu sonuca ulaşırız:

```
X_train, X_test, y_train, y_test = train_test_split(X, s,
random_state=42)
steps = [('normalizer', Normalizer()),
('scaler', StandardScaler()),
('knn', KNeighborsRegressor())]
pipeline = Pipeline(steps)
param_grid = {'knn__n_neighbors': np.arange(1, 8)}
cv = GridSearchCV(pipeline, param_grid)
cv.fit(X_train, y_train)
y_pred = cv.predict(X_test)
print('En iyi parametreler: {}'.format(cv.best_params_))
print('R^2: {}'.format(cv.score(X_test, y_test)))
print('np.mean(R^2): {}'.format(mean_squared_error(y_test,
y_pred)))
```

Çıktı:

```
En iyi parametreler: {'knn__n_neighbors': 2}
R^2: 0.9438547673940144
np.mean(R^2): 2.4539014474879512
```

En iyi modeli kaydetmek:

```
s_best = cv.best_estimator_
```

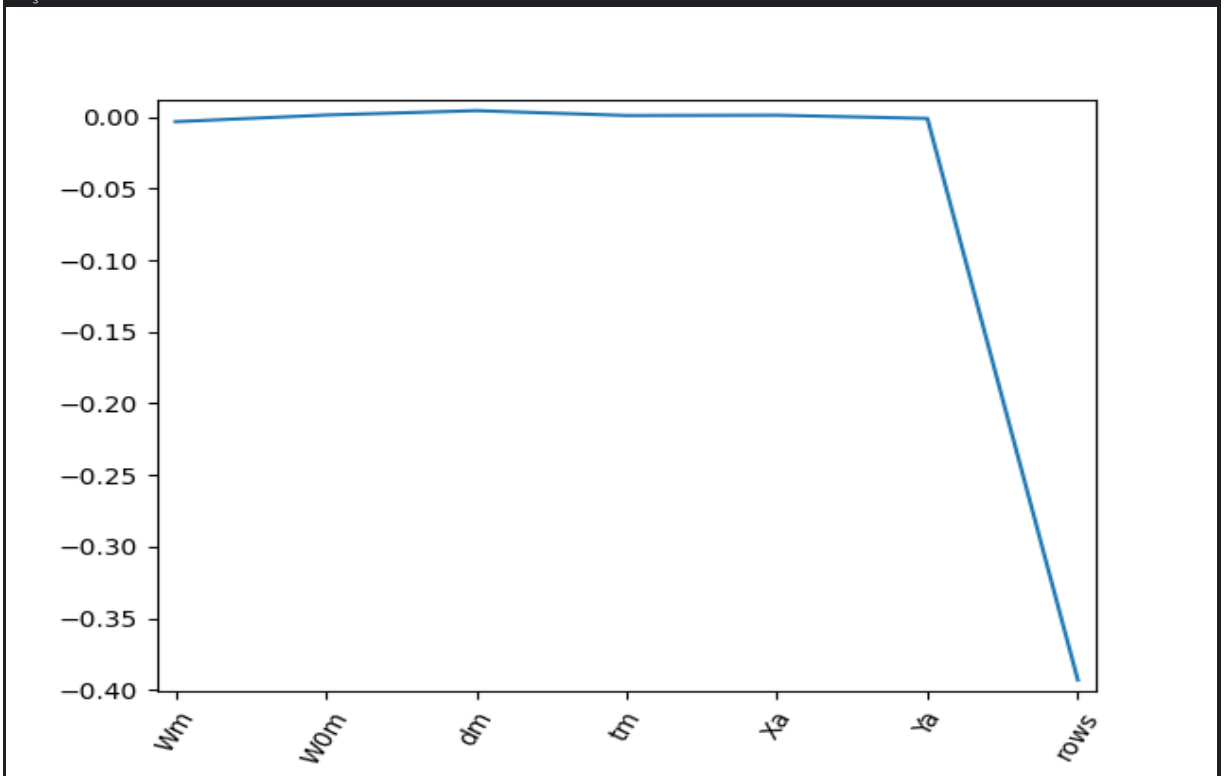
f) Bant Genişliği için Bir Model Oluşturma

Bant genişliğini y'den ayıralım.

```
bandwidth = y['bandwidth']

#Lasso regresyon algoritması aracılığıyla hangi özelliğin bant
genişliği üzerinde daha fazla etkiye sahip olduğunu kontrol
ediyoruz:
#Lasso Regresyonu üzerinde hiperparametre ayarı gerçekleştiriyoruz.
param_grid = {'alpha': np.arange(1e-4, 1e-3, 1e-4)}
lasso = Lasso(True)
lasso_cv = GridSearchCV(lasso, param_grid, cv=5)
lasso_cv.fit(X, bandwidth)

#En iyi alpha'yı ayarlıyoruz.
print('geri donus kaybi Lasso için en iyi alpha {}
dır'.format(lasso_cv.best_params_['alpha']))
lasso.alpha = lasso_cv.best_params_['alpha']
lasso.fit(X, bandwidth)
```

Çıktı:
bant genişliği Lasso için en iyi alpha 0.0009000000000000001 dır.

```
lasso_coef = lasso.coef_
plt.plot(range(len(X_columns)), lasso_coef)
plt.xticks(range(len(X_columns)), X_columns, rotation=60)
plt.margins(0.02)
plt.show()
#çıktı
```



```
X_train, X_test, y_train, y_test = train_test_split(X,
bandwidth,
random_state=42)
steps = [('normalizer', Normalizer()),
('scaler', StandardScaler()),
('knn', KNeighborsRegressor())]
pipeline = Pipeline(steps)
param_grid = {'knn__n_neighbors': np.arange(1, 8)}
cv = GridSearchCV(pipeline, param_grid)

cv.fit(X_train, y_train)
y_pred = cv.predict(X_test)
print('En iyi parametreler: {}'.format(cv.best_params_))
print('R^2: {}'.format(cv.score(X_test, y_test)))
print('np.mean(R^2): {}'.format(mean_squared_error(y_test,
y_pred)))
```

Çıktı:

```
En iyi parametreler: {'knn__n_neighbors': 2}
R^2: 0.8781878882090124
np.mean(R^2): 15.612865834843808
```

En iyi modeli kaydetmek:

```
bandwidth_best = cv.best_estimator_
```