Final Proje İçerik Yapısı:

Projenin Adı: Python' da Basit ve Çoklu Regresyon (İstanbul Su, Nüfus, Baraj Verisi)

Grup Üyeleri: Kaan Hacıömeroğlu, Halil İbrahim Altuntaş, Furkan Kütük, Rabia Gizem Kahraman, Fatma Gevrek

ÖZET

Python' da Basit ve Çoklu Regresyon yaparken 3 farklı veri kullandık. Bu veriler 2009-2019 yılları arasındaki "İstanbul'a verilen Temiz Su Miktarı, İstanbul Baraj Doluluk Oranları, İstanbul Nüfusu." Amacımız bunların arasındaki ilişkileri incelemektir.

Giriş

Python' da Basit ve Çoklu Regresyon yaparken 3 farklı veri kullandık. Bu veriler 2009-2019 yılları arasındaki "İstanbul'a verilen Temiz Su Miktarı, İstanbul Baraj Doluluk Oranları, İstanbul Nüfusu."

Vaka çalışmamızda öncelik olarak su datasını işledik. Daha sonrasında suyu etkileyen nüfus ve baraj doluluk oranlarını inceledik.

2009 ve 2019 arasında değerlendirme

10 yıl içinde verilen temiz su miktarındaki artış ve azalışlar,

Bu verilerin aylık ve yıllık olarak grafikleri,

İstanbul temiz su dağıtımlarını mevsimlere göre inceledik. İlkbahar, yaz, sonbahar, kış karşılaştırmalarını grafikleştirip yorumladık,

Diğer faktörler ile basit ve çoklu regresyon yaptık.

Veri Anlama

Veri Seti:

" https://www.nufusu.com/il/istanbul-nufusu "

"https://data.ibb.gov.tr/"

"https://www.iski.istanbul/web"

Veri Yapısı:

Veri setimiz text ve numeric değerlerden oluşmakta,

Eksik ve Aykırı değerler bulunmamaktadır.

VERİYİ HAZIRLAMA

Ön İşlemler:

Pandas kütüphanesini çalıştırdık. Verilerimizi okuttuk. Satır isimlerini değiştirdik. Ay sütununu indeks değer olarak atadık.

```
import pandas as pd
import numpy as np
pip install openpyxl
import openpyxl
veriSeti = pd.read_excel('istanbul_veri.xlsx')

veriSeti.index= ["Ocak", "Şubat", "Mart", "Nisan","Mayıs","Haziran","Temmuz","Ağustos","Eylül","Ekim","Kasım","Aralık" ]
del veriSeti["Ay"]
```

veriSeti - DataFrame

Index	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Ocak	55926	60626	65178	67137	71538	74528	77053	80206	81757	84212	83576
Şubat	50838	55607	59159	63613	64379	66844	67912	74114	73515	75692	74797
Mart	57261	60690	65895	67727	71663	73786	75459	79147	81215	84274	84666
Nisan	56034	60060	63179	66769	70859	72909	74177	80340	80663	84239	84968
Mayıs	62878	68131	69283	71704	80496	79584	82905	84951	87441	90369	93692
Haziran	65211	65795	72126	77507	79433	80020	82472	88033	87008	90831	93135
Temmuz	66197	68314	76919	83574	84224	84905	88453	88076	93177	95090	96028
Ağustos	66583	75594	75957	80098	82987	83262	89859	91773	91794	93402	94171
Eylül	61154	66734	71906	76396	80163	78215	84083	84767	87630	88053	91987
Ekim	61555	66419	70069	75991	76077	77767	82263	85454	87114	86864	90439
Kasım	59452	64087	65797	70456	73246	75344	79387	80802	84163	83213	86747
Aralık	60567	65969	68282	71964	74388	77285	81133	80958	85165	84728	51278

2009-20019 Yılları arasındaki toplam verilen su miktarı ve nüfus verisini okuttuk.

```
veriSeti2 = pd.read_excel('veri2.xlsx')
veriSeti2.index= ["2009", "2010", "2011","2012","2013","2014","2015","2016","2017","2018","2019"]
```

veriSeti2 - DataFrame

Index	Su Miktarı	Nüfus
2009	723656	12915158
2010	778026	13255685
2011	823750	13624240
2012	872936	13854740
2013	909453	14160467
2014	924449	14377018
2015	965156	14657434
2016	998621	14804116
2017	1020642	15029231
2018	1040967	15067724
2019	1025484	15519267

Veri setimizin özet bilgisini aldık.

```
#Özet Bilgileri Verir
ozetVeri = veriSeti.describe()
veriSeti.describe()
```

ozetVeri - DataFrame

Index	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
count	12	12	12	12	12	12	12	12	12	12	12
mean	60304.7	64835.5	68645.8	72744.7	75787.8	77037.4	80429.7	83218.4	85053.5	86747.2	85457
std	4711.73	5138.61	5176.85	6034.34	5843.62	4849.48	6133.16	4844.21	5314.22	5248.57	12320.8
min	50838	55607	59159	63613	64379	66844	67912	74114	73515	75692	51278
25%	56954.2	60674	65642.2	67579.5	71631.8	74342.5	76654.5	80306.5	81621.5	84232.2	84393.5
50%	60860.5	65882	68782.5	71834	75232.5	77526	81698	82862.5	86086.5	85796	88593
75%	63461.2	67083.2	71961	76673.8	80246.2	79693	83199.5	86098.8	87488.2	90484.5	93274.2
max	66583	75594	76919	83574	84224	84905	89859	91773	93177	95090	96028

Eksik verileri bulduk.

```
# Sutunlardaki eksik verinin kac adet oldugunun bulunmasi
print(veriSeti.isnull().sum())
```

```
In [12]: print(veriSeti.isnull().sum())
2009    0
2010    0
2011    0
2012    0
2013    0
2014    0
2015    0
2016    0
2017    0
2018    0
2019    0
dtype: int64
```

Veri setini normalize ettik.

```
#Normalize
n_veriSeti = (veriSeti - np.min(veriSeti))/(np.max(veriSeti)-np.min(veriSeti))
```

n_veriSeti - DataFrame

Index	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Ocak	0.32315	0.251113	0.338908	0.176544	0.360746	0.425447	0.416503	0.34498	0.419184	0.439221	0.721743
Şubat	0	0	0	0	0	0	0	0	0	0	0.525564
Mart	0.407939	0.254315	0.379279	0.206102	0.367045	0.384364	0.343874	0.28501	0.391618	0.442417	0.746101
Nisan	0.33001	0.222795	0.226351	0.158108	0.326531	0.335806	0.28546	0.352568	0.363544	0.440612	0.752849
Mayıs	0.764687	0.626607	0.570045	0.40534	0.812144	0.705387	0.683146	0.613681	0.70827	0.756624	0.947799
Haziran	0.912861	0.509731	0.730124	0.696057	0.758579	0.729528	0.663416	0.78821	0.686248	0.780441	0.935352
Temmuz	0.975484	0.635763	1	1	1	1	0.935937	0.790645	1	1	1
Ağustos	1	1	0.945833	0.82586	0.937667	0.909031	1	1	0.929661	0.912981	0.958503
Eylül	0.655192	0.556712	0.717736	0.640399	0.795364	0.629589	0.736821	0.603262	0.717882	0.637231	0.909698
Ekim	0.680661	0.540952	0.614302	0.620109	0.589468	0.604784	0.653893	0.642165	0.691639	0.575936	0.875106
Kasım	0.547094	0.424276	0.373761	0.342818	0.446813	0.470627	0.522851	0.37873	0.541552	0.38772	0.792603
Aralık	0.61791	0.518437	0.513682	0.418366	0.504359	0.578096	0.602406	0.387564	0.592513	0.465821	0

Gruplandırma:

Loc ve İloc kullanarak Aylara göre İlkbahar, Yaz, Sonbahar, Kış olarak mevsimlere ayırdık.

```
#Mevsimlere Ayırma
ilkbahar = veriSeti[2:5]
yaz = veriSeti[5:8]
sonbahar = veriSeti[8:11]
kis=veriSeti.iloc[[11,0,1],]
```

sonbahar - DataFrame

Index	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Eylül	61154	66734	71906	76396	80163	78215	84083	84767	87630	88053	91987
Ekim	61555	66419	70069	75991	76077	77767	82263	85454	87114	86864	90439
Kasım	59452	64087	65797	70456	73246	75344	79387	80802	84163	83213	86747

yaz - DataFrame

Index	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Haziran	65211	65795	72126	77507	79433	80020	82472	88033	87008	90831	93135
Temmuz	66197	68314	76919	83574	84224	84905	88453	88076	93177	95090	96028
Ağustos	66583	75594	75957	80098	82987	83262	89859	91773	91794	93402	94171

kis - DataFrame

Index	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Aralık	60567	65969	68282	71964	74388	77285	81133	80958	85165	84728	51278
Ocak	55926	60626	65178	67137	71538	74528	77053	80206	81757	84212	83576
Şubat	50838	55607	59159	63613	64379	66844	67912	74114	73515	75692	74797

ilkbahar - DataFrame

Index	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Mart	57261	60690	65895	67727	71663	73786	75459	79147	81215	84274	84666
Nisan	56034	60060	63179	66769	70859	72909	74177	80340	80663	84239	84968
Mayıs	62878	68131	69283	71704	80496	79584	82905	84951	87441	90369	93692

Hangi mevsimde daha çok kullanıldığını bulmak için mevsim ortalamalarını bulduk.

```
#Mevsim Ortalamaları Bulma
ilkbaharort = ilkbahar.mean()
round(ilkbaharort.mean(),2)
ilkbaharort = ilkbaharort.to_frame()
ilkbaharort.rename(columns={0: 'İlk Bahar'}, inplace=True)

yazort = yaz.mean()
round(yazort.mean(),2)
yazort = yazort.to_frame()
yazort.rename(columns={0: 'Yaz'}, inplace=True)

sonbaharort = sonbahar.mean()
round(sonbaharort.mean(),2)
sonbaharort = sonbaharort.to_frame()
sonbaharort.rename(columns={0: 'Son Bahar'}, inplace=True)

kisort = kis.mean()
round(kisort.mean(),2)
kisort = kisort.to_frame()
kisort.rename(columns={0: 'Kış'}, inplace=True)

#Mevsim Ortalamalarını Birleştirme
mevsimler = pd.concat([ilkbaharort,yazort,sonbaharort,kisort], join = "outer", axis=1)
```

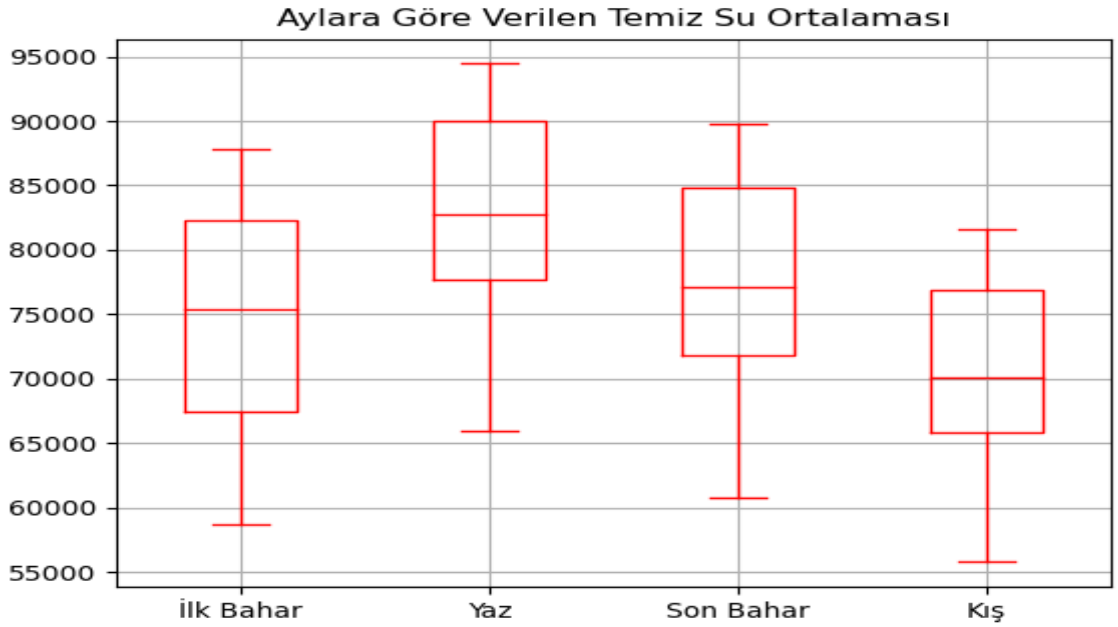
mevsimler - DataFrame

Index	İlk Bahar	Yaz	Son Bahar	Kış
2009	58724.3	65997	60720.3	55777
2010	62960.3	69901	65746.7	60734
2011	66119	75000.7	69257.3	64206.3
2012	68733.3	80393	74281	67571.3
2013	74339.3	82214.7	76495.3	70101.7
2014	75426.3	82729	77108.7	72885.7
2015	77513.7	86928	81911	75366
2016	81479.3	89294	83674.3	78426
2017	83106.3	90659.7	86302.3	80145.7
2018	86294	93107.7	86043.3	81544
2019	87775.3	94444.7	89724.3	69883.7

Görselleştirme:

```
#Aylara Göre Verilen Temiz Su Ortalaması (PlotBox)
pip install matplotlib
import matplotlib.pyplot as plt

mevsimler.plot.box(grid='True', color="red")
plt.title("Aylara Göre Verilen Temiz Su Ortalaması")
plt.show()

#Çizgi grafiği
plt.plot (ilkbahar, "o:r")
plt.title("İlk Bahar")

plt.plot (yaz, "o:r")
plt.title("Yaz")

plt.plot (sonbahar, "o:r")
plt.title("Son Bahar")

plt.plot (kis, "o:r")
plt.title("Kış")
```

Boxplot grafiğimize göre en çok temiz su miktarının yazın verildiğini, en az verilen temiz su miktarının ise kışın verildiğini gözlemlemekteyiz. Bahar aylarında ise verilen su miktarının birbirine yakın olduğu görülüyor. Buna göre yaz mevsiminde suyu daha bilinçli kullanarak tasarruf sağlayabiliriz.



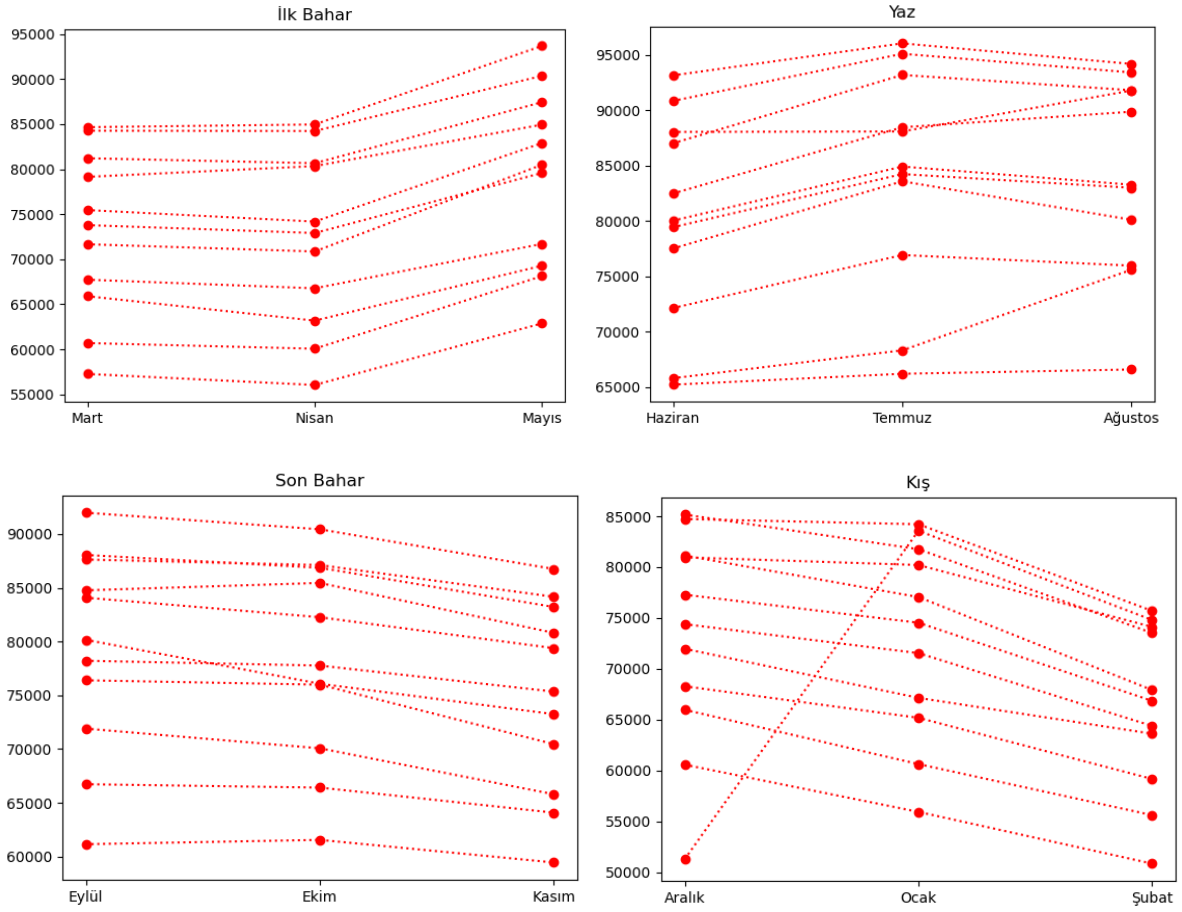
Aşağıdaki grafiklerde mevsimleri içeren aylara göre kullanılan su miktarının çizgi grafiğini görüyoruz.

İlkbahar mevsiminde nisan ayından sonra verilen su miktarında her yıl nisan ve mayıs ayları arasında artış görülmektedir.

Yaz mevsiminde genellikle temmuz ayında hep max seviyede su verilmiştir.

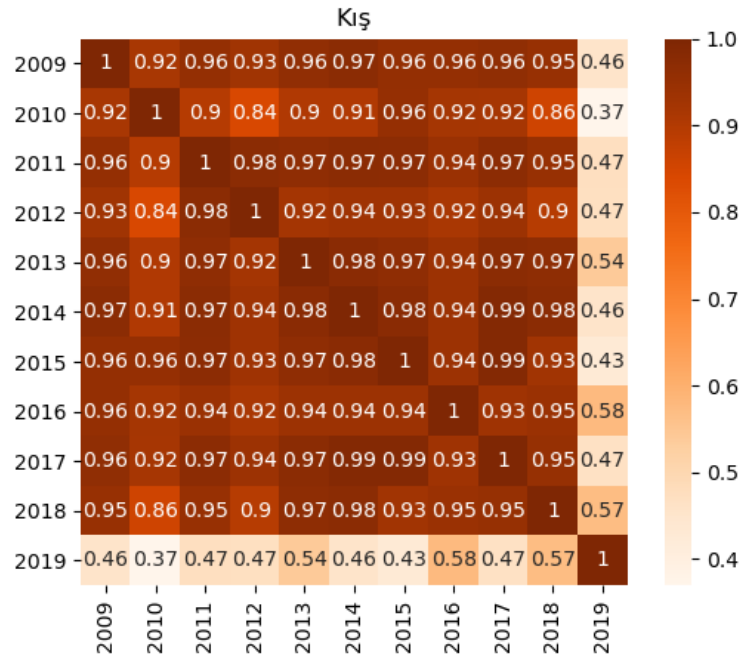
Sonbahar Eylül ve Ekim ayları arasında verilen su miktarı daha stabil ilerlerken Kasım ayına doğru azalış görülmektedir.

Kış mevsiminden ilkbahara geçişte verilen temiz su miktarı azalış göstermektedir. Fakat 2019 yılında verilen su miktarı 50000 civarında iken ocak ayında 83000'e fırladığını görebiliriz.



Rengin koyuluğu korelasyon derecesinin daha güçlü olduğunu belirtmektedir. Korelasyon + yönde olduğunda değişkenler arasında pozitif yönlü bir ilişki vardır, bu değerin 1'e yakın değişkenler arasında doğrusal bir ilişki vardır yorumu da yapılabilir.

```
#Isı haritası
import seaborn as sns
corr = n_veriSeti.corr()
sns.heatmap(
    corr,
    annot = True, # Korelasyon degerlerinin grafigin uzerine yazdirma
    square=True, # Kutularin kare bicimde gosterilmesi
    cmap="Oranges" # Renklendirme secenegi
)
```

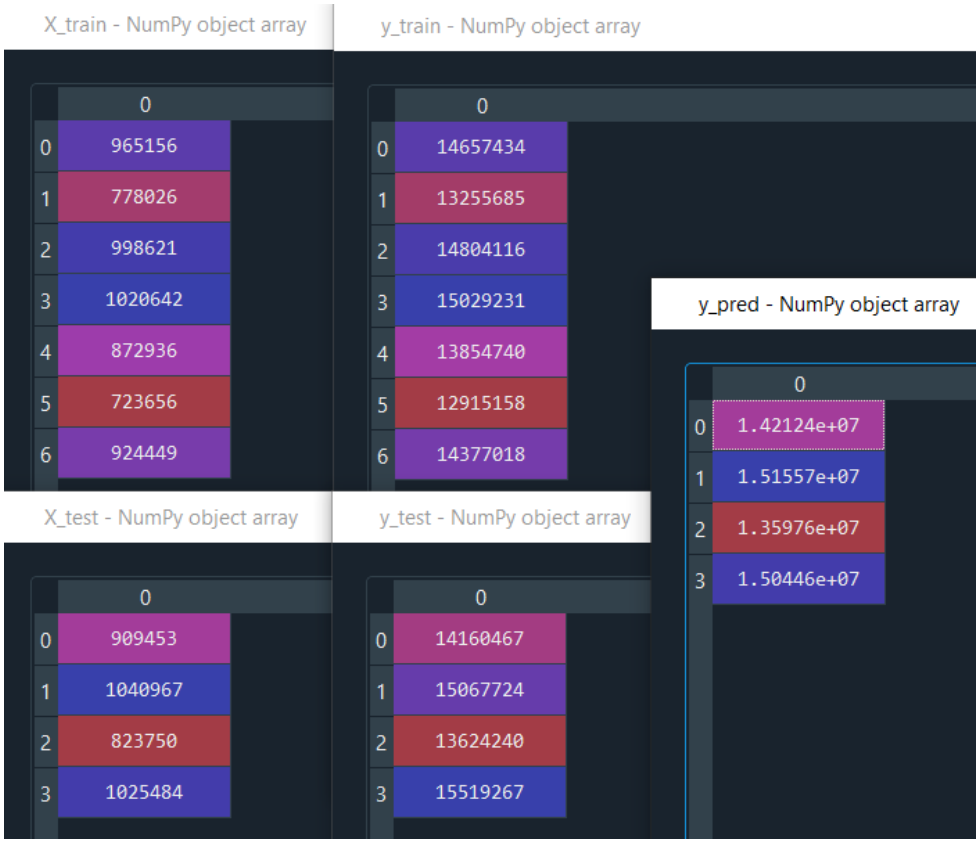


MODELLEME

Basit Regresyon:

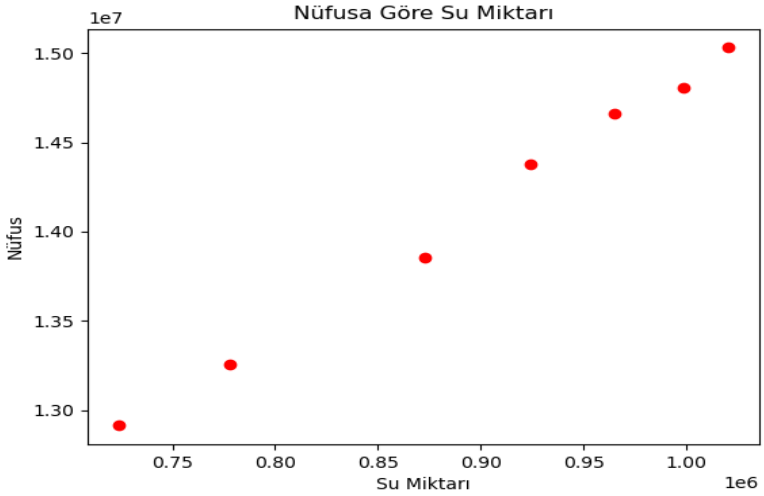
VeriSeti2'nin Bağımlı ve Bağımsız değişkenlerini atadık. Eğitim ve Test verilerini oluşturup, bilgisayara Model üzerinden tahmin yaptırdık.

```
#BASİT REGRESYON

#bağımlı ve bağımsız değişken oluşturma
X = veriSeti2.iloc[:, :-1].values
y = veriSeti2.iloc[:, 1].values

#Eğitim ve Test veriseti oluşturma
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)

#Regresyon Modelini Eğitme
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

#Model üzerinden  Tahmin  yaptırma
y_pred = regressor.predict(X_test)
```



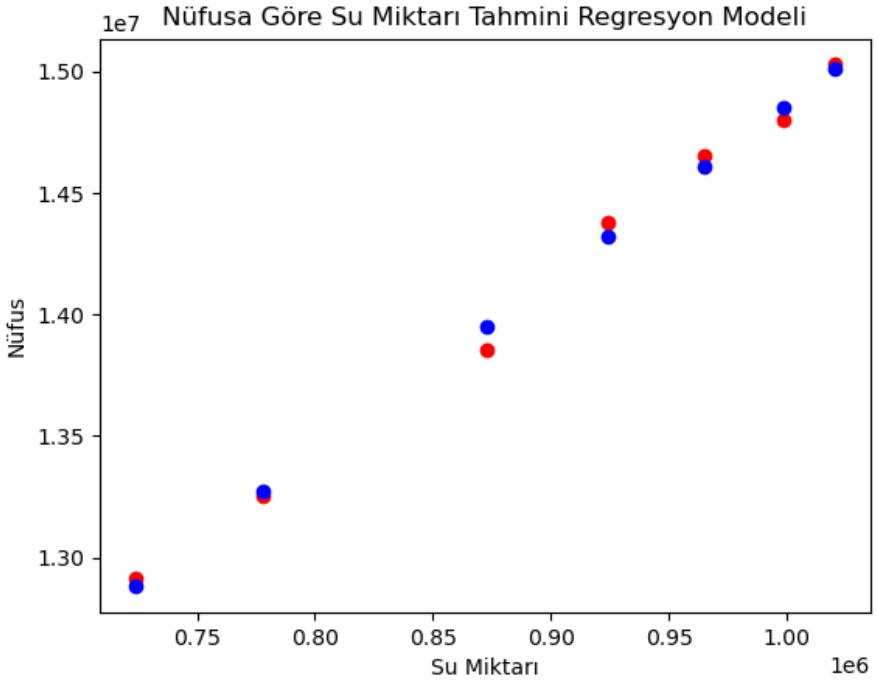
Eğitim veri setine ait saçılma diyagramını oluşturduk.

```
#REGRESYON GRAFİĞİ
#eğitim veri setine ait saçılma diyagramı
plt.scatter(X_train,y_train, color = 'red')
plt.title('Nüfusa Göre Su Miktarı')
plt.xlabel('Su Miktarı')
plt.ylabel('Nüfus')
plt.show()
```



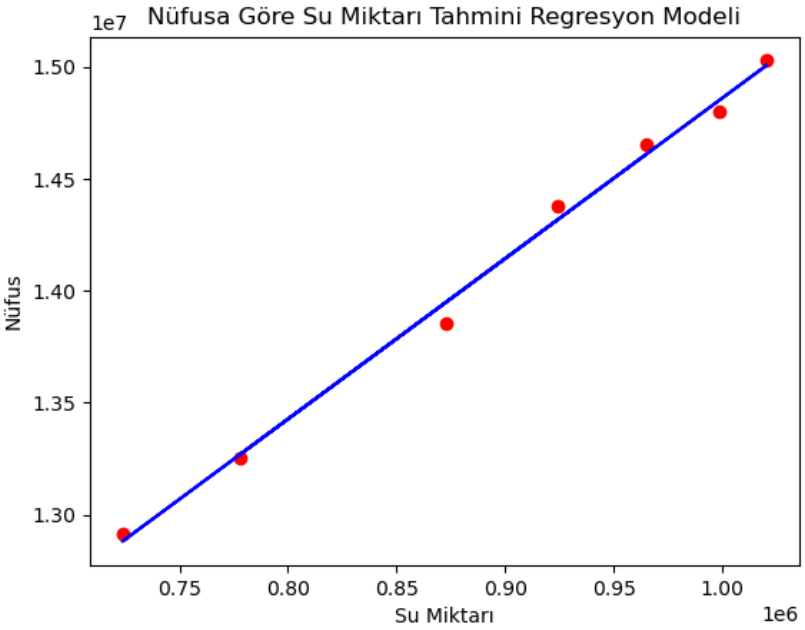
Eğitim veri setinin üstüne tahmin veri setini ekleyerek saçılma diyagramını oluşturduk.

```
#tahmin veri setine ait saçılma diyagramı
plt.scatter(X_train, y_train, color = 'red')
modelin_tahmin_ettigi_y = regressor.predict(X_train)
plt.scatter(X_train, modelin_tahmin_ettigi_y, color = 'blue')
plt.title('Nüfusa Göre Su Miktarı Tahmini Regresyon Modeli')
plt.xlabel('Su Miktarı')
plt.ylabel('Nüfus')
plt.show()
```



Doğrusal çizgi üzerine önceki grafiğimizi ekledik.

```
plt.scatter(X_train, y_train, color = 'red')
modelin_tahmin_ettigi_y = regressor.predict(X_train)
plt.plot(X_train, modelin_tahmin_ettigi_y, color = 'blue')
plt.title('Nüfusa Göre Su Miktarı Tahmini Regresyon Modeli')
plt.xlabel('Su Miktarı')
plt.ylabel('Nüfus')
plt.show()
```



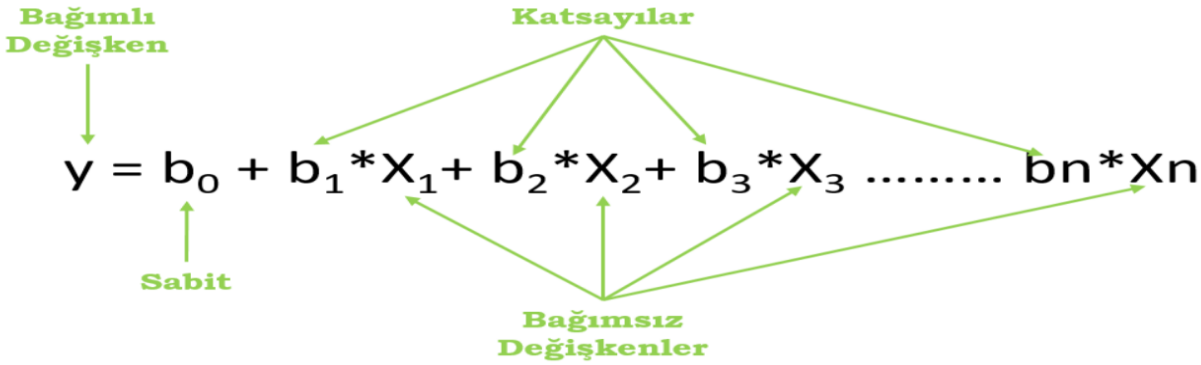
Bu saçılma diyagramlarına göre nüfus ve verilen su miktarı arasındaki oran doğrusal olarak artış görülmektedir. Eğitim ve tahmin veri setlerimiz birbirine yakın olduğunu gözlemliyoruz.

Çoklu Regresyon:

Bir olayın sonuç olarak doğmasına sebep olan faktörler genelde birden fazladır.Onun için regresyon modelinde bir bağımlı değişken ile birden fazla bağımsız değişkenin ilişkisi aranabilir.

Bağımlı Değişkenimiz; “Yıllar”

Bağımsız Değişkenimiz; “Nüfus, Baraj Doluluk oranı, Su Miktarı ve Yıllar”



VeriSeti2'ye Baraj Doluluk Oranlarını ve Yılları ekledik.

```
#veri setine ek veriler giriliyor
veriSeti2["Baraj Doluluk Oranları m^3"] = pd.Series(["35","93","83","60","64","35","70","58","53","64","83"],index=veriSeti2.index)
veriSeti2["Yıllar"] = pd.Series(["2009","2010","2011","2012","2013","2014","2015","2016","2017","2018","2019"],index=veriSeti2.index)
```

veriSeti2 - DataFrame

Index	Su Miktarı	Nüfus	Baraj Doluluk Oranları m^3	Yıllar
2009	723656	12915158	35	2009
2010	778026	13255685	93	2010
2011	823750	13624240	83	2011
2012	872936	13854740	60	2012
2013	909453	14160467	64	2013
2014	924449	14377018	35	2014
2015	965156	14657434	70	2015
2016	998621	14804116	58	2016
2017	1020642	15029231	53	2017
2018	1040967	15067724	64	2018
2019	1025484	15519267	83	2019

Bağımlı ve Bağımsız değişken atadık. Eğitim ve Test verilerini oluşturduk.

```
#bağımlı ve bağımsız değişken oluşturma
xx = veriSeti2.iloc[:, :-1].values
yy = veriSeti2.iloc[:, 3].values
xx = xx.astype(int)
yy = yy.astype(int)

#Eğitim ve Test veriseti oluşturma
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_xx = LabelEncoder()
xx[:, 1] = labelencoder_xx.fit_transform(xx[:, 1])


from sklearn.model_selection import train_test_split
xx_train, xx_test, yy_train, yy_test = train_test_split(xx, yy, test_size = 0.2, random_state = 0)
```

xx_test - NumPy object array

	0	1	2
0	909453	4	64
1	1040967	9	64
2	823750	2	83

yy_test - NumPy object array

	0
0	2013
1	2018
2	2011

xx_train - NumPy object array

	0	1	2
0	1025484	10	83
1	965156	6	70
2	778026	1	93
3	998621	7	58
4	1020642	8	53
5	872936	3	60
6	723656	0	35
7	924449	5	35

yy_train - NumPy object array

	0
0	2019
1	2015
2	2010
3	2016
4	2017
5	2012
6	2009
7	2014

Regresyon modelimize linear regresyon fonksiyonunu kullanarak eğittik ve bilgisayara model üzerinden tahmin yaptırdık.

```
#Regresyon Modelini Eğitme
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(xx_train, yy_train)

#Model üzerinden  Tahmin  yaptırma
yy_pred = regressor.predict(xx_test)
```

yy_pred - NumPy object array

	0
0	2013
1	2018
2	2011

Tablomuzun başına sabit değişken ekledik. 0. indeks sabit değişkenimiz.

```
#Sabit değişkeni ekleme
xx = np.append(arr = np.ones((11,1)).astype(int), values = xx, axis = 1)
```

xx - NumPy object array

	0	1	2	3
0	1	723656	0	35
1	1	778026	1	93
2	1	823750	2	83
3	1	872936	3	60
4	1	909453	4	64
5	1	924449	5	35
6	1	965156	6	70
7	1	998621	7	58
8	1	1020642	8	53
9	1	1040967	9	64
10	1	1025484	10	83

Çoklu Regresyon modelini kurduk.

```
import statsmodels.api as sm

#Birinci tur
xx_opt = xx[:, [0,1,2,3]]
regressor_OLS = sm.OLS(endog=yy, exog=xx_opt).fit()
regressor_OLS.summary()
```

Çıkarma kuralımız en yüksek p değerine sahip değişkeni modelden çıkarmak idi. P değerleri sütununda X3'ün en yüksek p değerine (0.984) sahip olduğunu görüyoruz, öyleyse X3'ü çıkarıyoruz.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                1.146e+19
Date:                Sat, 03 Apr 2021   Prob (F-statistic):           8.86e-66
Time:                        18:58:44   Log-Likelihood:                 208.44
No. Observations:                  11   AIC:                            -408.9
Df Residuals:                       7   BIC:                            -407.3
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       2009.0000   1.88e-08   1.07e+11      0.000    2009.000    2009.000
x1         -1.675e-15   2.43e-14     -0.069      0.947    -5.9e-14    5.57e-14
x2             1.0000   7.83e-10   1.28e+09      0.000       1.000       1.000
x3          6.217e-13   3.08e-11      0.020      0.984   -7.23e-11    7.35e-11
==============================================================================
Omnibus:                        1.516   Durbin-Watson:                   0.001
Prob(Omnibus):                  0.469   Jarque-Bera (JB):                1.104
Skew:                          -0.598   Prob(JB):                        0.576
Kurtosis:                       2.011   Cond. No.                     3.22e+07
==============================================================================
```

X3'ü devre dışı bırakarak modeli tekrar çalıştırıp özetini alıyoruz.

```
#İkinci tur
xx_opt = xx[:, [0,1,2]]
regressor_OLS = sm.OLS(endog=yy, exog=xx_opt).fit()
regressor_OLS.summary()
```

Tekrar Çıkarma kuralını uyguluyoruz. En yüksek p değerine sahip değişkeni X1'i (0.948) sahip olduğunu görüyoruz, öyleyse X1'i çıkarıyoruz.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                1.199e+21
Date:                Sat, 03 Apr 2021   Prob (F-statistic):           1.24e-82
Time:                        19:09:08   Log-Likelihood:                231.05
No. Observations:                  11   AIC:                           -456.1
Df Residuals:                       8   BIC:                           -454.9
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       2009.0000   2.19e-09   9.19e+11      0.000    2009.000    2009.000
x1         -1.917e-16   2.87e-15     -0.067      0.948   -6.82e-15    6.44e-15
x2             1.0000   9.26e-11   1.08e+10      0.000       1.000       1.000
==============================================================================
Omnibus:                        1.143   Durbin-Watson:                   0.001
Prob(Omnibus):                  0.565   Jarque-Bera (JB):                0.884
Skew:                          -0.480   Prob(JB):                        0.643
Kurtosis:                       1.996   Cond. No.                     3.12e+07
==============================================================================
```

X1'i devre dışı bırakarak modeli tekrar çalıştırıp özetini alıyoruz.

```
#Üçüncü tur
xx_opt = xx[:, [0,2]]
regressor_OLS = sm.OLS(endog=yy, exog=xx_opt).fit()
regressor_OLS.summary()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                5.039e+26
Date:                Sat, 03 Apr 2021   Prob (F-statistic):          3.52e-117
Time:                        19:13:04   Log-Likelihood:                297.81
No. Observations:                  11   AIC:                            -591.6
Df Residuals:                       9   BIC:                            -590.8
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       2009.0000   2.64e-13   7.62e+15      0.000    2009.000    2009.000
x1             1.0000   4.45e-14   2.24e+13      0.000       1.000       1.000
==============================================================================
Omnibus:                        9.711   Durbin-Watson:                   0.026
Prob(Omnibus):                  0.008   Jarque-Bera (JB):                5.230
Skew:                          -1.650   Prob(JB):                       0.0732
Kurtosis:                       3.722   Cond. No.                         11.3
==============================================================================
```

Evet, en sonunda p değerlerini 0.05'in altında bırakmayı başardık. Geriye sabit ve yıllar kaldı.