In [1]:

```python
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

In [2]:

```python
veri=pd.read_csv("dataset/DogrusalRegresyon.csv")
```

In [3]:

```python
veri.head()
```

Out[3]:

	Cinsiyet	Yas_Araligi	Bas_cevresi(cm^3)	Beyin_agirligi(gr)
0	1	1	4512	1530
1	1	1	3738	1297
2	1	1	4261	1335
3	1	1	3777	1282
4	1	1	4177	1590

In [4]:

```python
X=veri['Bas_cevresi(cm^3)'].values
```

In [5]:

```python
y=veri['Beyin_agirligi(gr)']
```

In [6]:

```python
len(X)
```

Out[6]:

```
237
```

In [10]:

```python
X.shape
```

Out[10]:

```
(237, 1)
```

In [9]:

```python
X=X.reshape(len(X),1)
```

In [11]:

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=42)
```

In [13]:

```python
model = LinearRegression()
model.fit(X_train,y_train)
```

Out[13]:

```
LinearRegression()
```

In [14]:

```python
y_pred=model.predict(X_test)
```

In [15]:

```python
%matplotlib --
plt.scatter(X_train, y_train, color='red')
plt.plot(X_train, model.predict(X_train))
```

Out[15]:

```
Using matplotlib backend: Qt5Agg
[<matplotlib.lines.Line2D at 0x29d8a368fa0>]
```

In [17]:

```python
plt.scatter(X_train, y_train, color='red')
plt.plot(X_test, model.predict(X_test))
```

Out[17]:

```
[<matplotlib.lines.Line2D at 0x29d88b4f430>]
```

In [19]:

```python
print(model.coef_)
print(model.intercept_)
```

```
[0.26481251]
319.8301925400709
```

In [20]:

```python
from sklearn.metrics import explained_variance_score, mean_absolute_error,mean_squared_error, median_absolute_error, r2_score
```

In [21]:

```python
print(r2_score(y_test, y_pred))
print(mean_absolute_error(y_test, y_pred))
print(mean_squared_error(y_test, y_pred))
print(median_absolute_error(y_test, y_pred))
```

```
0.6993002108399291
55.18499937560606
4617.918994815832
53.33675115672645
```

In [28]:

```python
#polinom regresyon
```

In [18]:

```python
def fonk(x):
    return 10*np.sin(x/2) + x/7 +6
```

In [19]:

```python
X=np.random.rand(100)*10
```

In [20]:

```python
y=fonk(X) + 2*np.random.randn(*X.shape)
```

In [21]:

```python
X=X.reshape(len(X),1)
```

In [22]:

```python
plt.scatter(X, y, color='red')
```

Out[22]:

```
<matplotlib.collections.PathCollection at 0x1dbbadd4a30>
```

In [23]:

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=42)
```

In [24]:

```python
from sklearn.preprocessing import PolynomialFeatures
```

In [25]:

```python
model_pol=PolynomialFeatures(degree=3)
```

In [26]:

```python
X_pol=model_pol.fit_transform(X_train)
```

In [27]:

```python
model_pol.fit(X_pol, y_train)
```

Out[27]:

```
PolynomialFeatures(degree=3)
```

In [28]:

```python
model2=LinearRegression()
model2.fit(X_pol, y_train)
```

Out[28]:

```
LinearRegression()
```

In [29]:

```python
X_grid=np.arange(min(X_train), max(X_train), 0.1)
X_grid=X_grid.reshape(len(X_grid),1)
plt.scatter(X_train, y_train, color='red')
plt.plot(X_grid, model2.predict(model_pol.fit_transform(X_grid)), color='blue')
```

Out[29]:

```
[<matplotlib.lines.Line2D at 0x1dbbad87250>]
```

In [30]:

```python
model2.intercept_
model2.coef_
```

Out[30]:

```
array([ 0.        , 11.16507618, -2.49334386,  0.13277488])
```

In [31]:

```python
y_pred=model2.predict(model_pol.fit_transform(X_test))
```

In [32]:

```python
r2_score(y_test, y_pred)
```

Out[32]:

```
0.8783706312445749
```

In [45]:

```python
"""
Reshape your data either using array.reshape(-1, 1) if your data has a single feature
or array.reshape(1, -1) if it contains a single sample.
"""
```

Out[45]:

```
'\nReshape your data either using array.reshape(-1, 1) if your data has a single feature \nor array.reshape(1, -1) if it contains a single sample.\n'
```

In [82]:

```python
#çoklu doğrusal regresyon
```

In [33]:

```python
veriseti=pd.read_excel('dataset/pv.xlsx')
```

In [34]:

```python
veriseti
```

Out[34]:

	ALT	LAT	SSD	T	GSR	GUC	EKSEN	SONUC
0	50	36.99	7.48	17.7	1.81	1	1	1.716690
1	50	36.99	7.48	17.7	1.81	1	2	2.235164
2	50	36.99	7.48	17.7	1.81	1	3	2.341118
3	50	36.99	7.48	17.7	1.81	3	1	5.150090
4	50	36.99	7.48	17.7	1.81	3	2	6.705519
...	...	...	...	...	...	...	...	...
1615	124	41.45	5.90	12.9	1.52	5	2	9.189464
1616	124	41.45	5.90	12.9	1.52	5	3	9.660183
1617	124	41.45	5.90	12.9	1.52	10	1	14.411507
1618	124	41.45	5.90	12.9	1.52	10	2	18.378948
1619	124	41.45	5.90	12.9	1.52	10	3	19.320388

1620 rows × 8 columns

In [85]:

```python
import re
missed_data=[]
non_categorical=[]

for i in veriseti:
    essiz=veriseti[i].unique()
    print(f'{i} özniteliğine ait eşsiz  veriler {essiz.size}')

    if(True in pd.isnull(essiz)):
        s="{} özniteliğine ait kayıp veriler {}".format(i, pd.isnull(veriseti[i]).sum())
        missed_data.append(s)
    """
    for j in range(1, np.prod(essiz.shape)):
        if(re.match('nan', str(essiz[j]))):
            break
        if not (re.search('(^\d+\.?\d*$)|(^\d*\.?\d+$)', str(essiz[i]))):
            non_categorical.append(i)
            break"""
```

```
ALT özniteliğine ait eşsiz  veriler 84
LAT özniteliğine ait eşsiz  veriler 115
SSD özniteliğine ait eşsiz  veriler 114
T özniteliğine ait eşsiz  veriler 66
GSR özniteliğine ait eşsiz  veriler 43
GUC özniteliğine ait eşsiz  veriler 4
EKSEN özniteliğine ait eşsiz  veriler 3
SONUC özniteliğine ait eşsiz  veriler 1592
```

In [35]:

```python
X=veriseti.iloc[:,:-1].values
```

In [36]:

```python
y=veriseti.iloc[:,-1].values
```

In [37]:

```python
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=42)
```

In [38]:

```python
from scipy import stats
```

In [39]:

```python
desc=stats.describe(X_train, axis=0)
```

In [40]:

```python
#veri setinin normal dağılıma sahip olduğunu test etmek için skewnes ve kurtosis
#katsayıları kullınalabilir. skewness 3'ten küçük ve kurtosis -2 ile 2 arasındaise normal dağılım denir
desc.kurtosis
```

Out[40]:

```
array([-0.66747085, -0.38510403,  0.22268709, -0.88863592, -0.01031744,
       -1.13685944, -1.48765007])
```

In [41]:

```python
desc.skewness
```

Out[41]:

```
array([ 0.34999757, -0.13744134, -0.86040611, -0.10834814,  0.07685618,
        0.52700093, -0.01613518])
```

In [89]:

```python
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
X_train_scaled=scaler.fit_transform(X_train)
print("per-feature minimum before scaling:\n {}".format(X_train.min(axis=0)))
print("per-feature maximum before scaling:\n {}".format(X_train.max(axis=0)))
print("per-feature minimum after scaling:\n {}".format(X_train_scaled.min(axis=0)))
print("per-feature maximum after scaling:\n {}".format(X_train_scaled.max(axis=0)))
```

```
per-feature minimum before scaling:
 [ 0.   33.55  3.83  6.    1.3   1.    1.  ]
per-feature maximum before scaling:
 [2.173e+03 4.203e+01 8.430e+00 1.850e+01 2.050e+00 1.000e+01 3.000e+00]
per-feature minimum after scaling:
 [0. 0. 0. 0. 0. 0. 0.]
per-feature maximum after scaling:
 [1. 1. 1. 1. 1. 1. 1.]
```

In [90]:

```python
X_test=scaler.transform(X_test)
print("per-feature minimum after scaling:\n{}".format(X_test.min(axis=0)))
print("per-feature maximum after scaling:\n{}".format(X_test.max(axis=0)))
```

```
per-feature minimum after scaling:
[0. 0. 0. 0. 0. 0. 0.]
per-feature maximum after scaling:
[1. 1. 1. 1. 1. 1. 1.]
```

In [91]:

```python
"""
# scale the data using MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

"""
```

Out[91]:

```
'\n# scale the data using MinMaxScaler\nscaler = MinMaxScaler()\nscaler.fit(X_train)\nX_train_scaled = scaler.transform(X_train)\nX_test_scaled = scaler.transform(X_test)\n\n'
```

In [42]:

```python
import statsmodels.formula.api as sm
X=np.append(arr=np.ones((1620,1)).astype(int), values=X, axis=1)
```

In [151…

```python
#veriseti['0']= 1
#veriseti.insert(loc=0, column='A', value=1)
```

In [45]:

```python
modelReg_OLS=sm.ols(formula='SONUC ~ ALT + LAT + SSD + T + GSR + GUC + EKSEN', data=veriseti).fit()
```

In [46]:

```python
modelReg_OLS.summary()
```

Out[46]:

OLS Regression Results

Dep. Variable:	SONUC	**R-squared:**	0.966
Model:	OLS	**Adj. R-squared:**	0.966
Method:	Least Squares	**F-statistic:**	6600.
Date:	Sun, 19 Mar 2023	**Prob (F-statistic):**	0.00
Time:	23:55:50	**Log-Likelihood:**	-2579.5
No. Observations:	1620	**AIC:**	5175.
Df Residuals:	1612	**BIC:**	5218.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-17.5192	2.914	-6.013	0.000	-23.234	-11.804
ALT	0.0004	0.000	2.652	0.008	0.000	0.001
LAT	0.1662	0.053	3.130	0.002	0.062	0.270
SSD	-0.1098	0.052	-2.104	0.036	-0.212	-0.007
T	0.0544	0.034	1.595	0.111	-0.013	0.121
GSR	5.0869	0.445	11.433	0.000	4.214	5.960
GUC	1.8728	0.009	211.472	0.000	1.855	1.890
EKSEN	1.2456	0.036	34.335	0.000	1.174	1.317

Omnibus:	55.010	**Durbin-Watson:**	1.322
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	138.055
Skew:	0.105	**Prob(JB):**	1.05e-30
Kurtosis:	4.415	**Cond. No.**	1.04e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.04e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [156…

```python
modelReg_OLS.summary2()
```

Out[156…

Model:	OLS	Adj. R-squared:	0.966
Dependent Variable:	SONUC	AIC:	5175.0930
Date:	2023-03-19 22:47	BIC:	5218.2145
No. Observations:	1620	Log-Likelihood:	-2579.5
Df Model:	7	F-statistic:	6600.
Df Residuals:	1612	Prob (F-statistic):	0.00
R-squared:	0.966	Scale:	1.4215

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-8.7596	1.4568	-6.0130	0.0000	-11.6170	-5.9022
A	-8.7596	1.4568	-6.0130	0.0000	-11.6170	-5.9022
ALT	0.0004	0.0002	2.6523	0.0081	0.0001	0.0008
LAT	0.1662	0.0531	3.1302	0.0018	0.0621	0.2703
SSD	-0.1098	0.0522	-2.1038	0.0355	-0.2122	-0.0074
T	0.0544	0.0341	1.5946	0.1110	-0.0125	0.1214
GSR	5.0869	0.4449	11.4330	0.0000	4.2142	5.9596
GUC	1.8728	0.0089	211.4715	0.0000	1.8555	1.8902
EKSEN	1.2456	0.0363	34.3345	0.0000	1.1745	1.3168

Omnibus:	55.010	Durbin-Watson:	1.322
Prob(Omnibus):	0.000	Jarque-Bera (JB):	138.055
Skew:	0.105	Prob(JB):	0.000
Kurtosis:	4.415	Condition No.:	1026397530802545536

In [47]:

```python
veriseti=veriseti.drop(columns='T')
```

In [48]:

```python
modelReg_OLS=sm.ols(formula='SONUC ~ ALT + LAT + SSD + GSR + GUC + EKSEN', data=veriseti).fit()
modelReg_OLS.summary2()
```

Out[48]:

Model:	OLS	Adj. R-squared:	0.966
Dependent Variable:	SONUC	AIC:	5175.6463
Date:	2023-03-19 23:56	BIC:	5213.3775
No. Observations:	1620	Log-Likelihood:	-2580.8
Df Model:	6	F-statistic:	7692.
Df Residuals:	1613	Prob (F-statistic):	0.00
R-squared:	0.966	Scale:	1.4228

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-14.6569	2.2960	-6.3836	0.0000	-19.1604	-10.1534
ALT	0.0002	0.0001	3.7337	0.0002	0.0001	0.0003
LAT	0.1133	0.0415	2.7316	0.0064	0.0319	0.1947
SSD	-0.0923	0.0511	-1.8076	0.0709	-0.1924	0.0079
GSR	5.0828	0.4451	11.4186	0.0000	4.2097	5.9559
GUC	1.8729	0.0089	211.3721	0.0000	1.8555	1.8902
EKSEN	1.2457	0.0363	34.3193	0.0000	1.1745	1.3169

Omnibus:	54.238	Durbin-Watson:	1.321
Prob(Omnibus):	0.000	Jarque-Bera (JB):	135.948
Skew:	0.098	Prob(JB):	0.000
Kurtosis:	4.406	Condition No.:	82143

In [49]:

```python
veriseti=veriseti.drop(columns='SSD')
```

In [50]:

```python
modelReg_OLS=sm.ols(formula='SONUC ~ ALT + LAT + GSR + GUC + EKSEN', data=veriseti).fit()
modelReg_OLS.summary2()
```

Out[50]:

Model:	OLS	Adj. R-squared:	0.966
Dependent Variable:	SONUC	AIC:	5176.9245
Date:	2023-03-19 23:56	BIC:	5209.2656
No. Observations:	1620	Log-Likelihood:	-2582.5
Df Model:	5	F-statistic:	9217.
Df Residuals:	1614	Prob (F-statistic):	0.00
R-squared:	0.966	Scale:	1.4248

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-16.3960	2.0862	-7.8592	0.0000	-20.4879	-12.3040
ALT	0.0002	0.0001	3.8496	0.0001	0.0001	0.0003
LAT	0.1475	0.0370	3.9904	0.0001	0.0750	0.2199
GSR	4.9392	0.4383	11.2691	0.0000	4.0795	5.7989
GUC	1.8729	0.0089	211.2279	0.0000	1.8555	1.8903
EKSEN	1.2458	0.0363	34.2981	0.0000	1.1745	1.3170

Omnibus:	57.253	Durbin-Watson:	1.320
Prob(Omnibus):	0.000	Jarque-Bera (JB):	145.682
Skew:	0.114	Prob(JB):	0.000
Kurtosis:	4.451	Condition No.:	75187

In [ ]: