

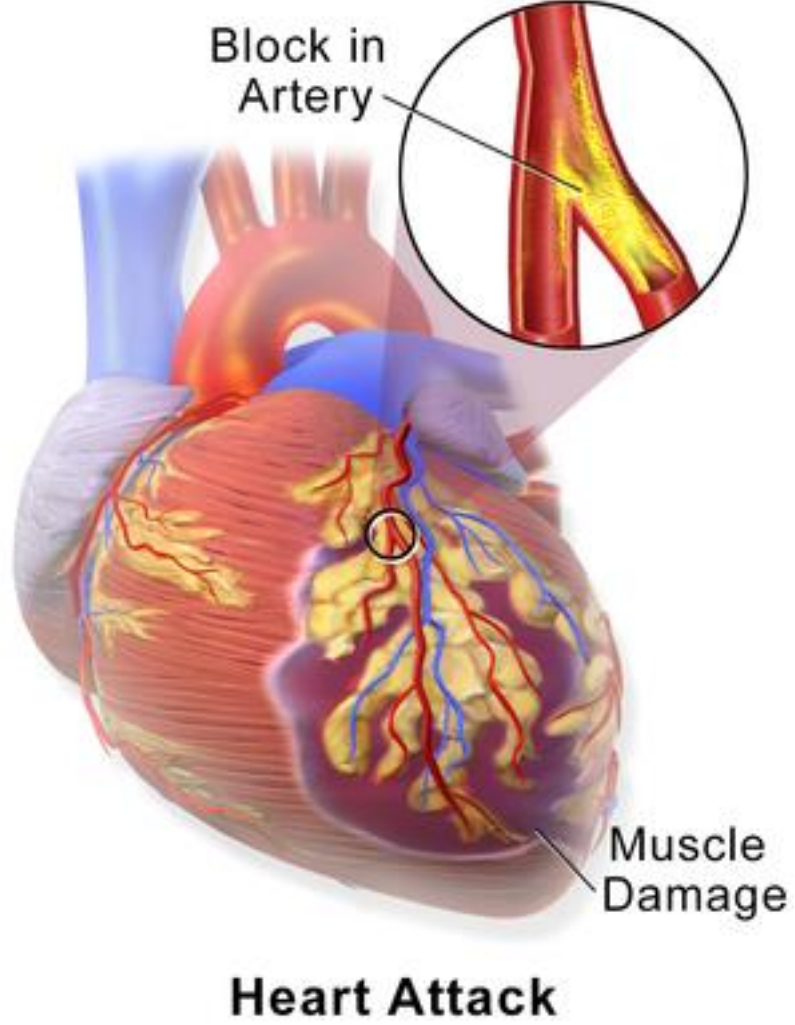


KSU Mhendislik ve Mimarlık Fakltesi Bilgisayar Mhendislięi

Semih ACAR

Adem YETER

Kalp krizi Nedir?



Kalp damarındaki plakların aniden yırtılması ve üzerine pıhtı oturması ile **kalp** damarı aniden tıkanabilir, sonuçta **kalp** kası oksijensiz kalır. Oksijensiz kalan **kalp** kası hücreleri bir süre sonra ölmeye başlar.

Projenin amacı

- Çalışmanın amacı, hastanın yaş, cinsiyet, göğüs ağrısı tipi, kan basıncı ve kolesterol gibi verilerinden faydalanarak kalp krizi riskini tahmin etmek ve analizde bulunmaktır.

Kalp krizi veri seti

- Age : Age of the patient
- Sex : Sex of the patient
- exang: exercise induced angina (1 = yes; 0 = no)
- ca: number of major vessels (0-3)
- cp : Chest Pain type chest pain type
 - Value 1: typical angina
 - Value 2: atypical angina
 - Value 3: non-anginal pain
 - Value 4: asymptomatic

Yaş bilgisi

Cinsiyet bilgisi

Egzersize bağlı göğüs ağrısı

Florosopi ile renklendirilen
büyük kapların sayısı

Göğüs ağrısı tipleri:

1- Tipik angina pectoris.

2- Atipik angina pectoris.

3- Angina dışı göğüs ağrısı.

4- Asemptomatik

Kalp krizi veri seti

- trtbps : resting blood pressure (in mm Hg)
- chol : cholestoral in mg/dl fetched via BMI sensor
- fbs : (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- rest_ecg : resting electrocardiographic results
 - Value 0: normal
 - Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
 - Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
- thalach : maximum heart rate achieved
- target : 0= less chance of heart attack 1= more chance of heart attack

Dinlenme kan basıncı

Serum kolestoral

Açlık kan şekeri

İstirahat elektrokardiyografik sonuçları

Maksimum kalp atış hızı

0= daha az kalp krizi geçirme şansı 1= daha fazla kalp krizi geçirme şans

Index	age	sex	cp	trtbns	chol	fbs	restecg	thalachh	exng	oldpeak	slo	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1
10	54	1	0	140	239	0	1	160	0	1.2	2	0	2	1
11	48	0	2	130	275	0	1	139	0	0.2	2	0	2	1
12	49	1	1	130	266	0	1	171	0	0.6	2	0	2	1
13	64	1	3	110	211	0	0	144	1	1.8	1	0	2	1
14	58	0	3	150	283	1	0	162	0	1	2	0	2	1
15	50	0	2	120	219	0	1	158	0	1.6	1	0	2	1
16	58	0	2	120	340	0	1							
17	66	0	3	150	226	0	1							

```
data = pd.read_csv("heart.csv")
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 303 entries, 0 to 302
```

```
Data columns (total 14 columns):
```

#	Column	Non-Null Count	Dtype
0	age	303 non-null	int64
1	sex	303 non-null	int64
2	cp	303 non-null	int64
3	trtbps	303 non-null	int64
4	chol	303 non-null	int64
5	fbs	303 non-null	int64
6	restecg	303 non-null	int64
7	thalachh	303 non-null	int64
8	exng	303 non-null	int64
9	oldpeak	303 non-null	float64
10	slp	303 non-null	int64
11	caa	303 non-null	int64
12	thall	303 non-null	int64
13	output	303 non-null	int64

```
dtypes: float64(1), int64(13)  
memory usage: 33.3 KB
```

```
data.info()
```

```
In [4]: data.shape
```

```
Out[4]: (302, 14)
```

```
In [2]: data.describe()
```

```
Out[2]:
```

	age	sex	cp	...	caa	thall	output
count	302.00000	302.000000	302.000000	...	302.000000	302.000000	302.000000
mean	54.42053	0.682119	0.963576	...	0.718543	2.314570	0.543046
std	9.04797	0.466426	1.032044	...	1.006748	0.613026	0.498970
min	29.00000	0.000000	0.000000	...	0.000000	0.000000	0.000000
25%	48.00000	0.000000	0.000000	...	0.000000	2.000000	0.000000
50%	55.50000	1.000000	1.000000	...	0.000000	2.000000	1.000000
75%	61.00000	1.000000	2.000000	...	1.000000	3.000000	1.000000
max	77.00000	1.000000	3.000000	...	4.000000	4.000000	4.000000

```
[8 rows x 14 columns]
```

```
data.describe()
```

```
age : 41  
sex : 2  
cp : 4  
trtbps : 49  
chol : 152  
fbs : 2  
restecg : 3  
thalachh : 91  
exng : 2  
oldpeak : 40  
slp : 3  
caa : 5  
thall : 4  
output : 2
```

```
### benzersiz değer sayısı
```

```
sutunlar = data.columns
```

```
for sutun in sutunlar:
```

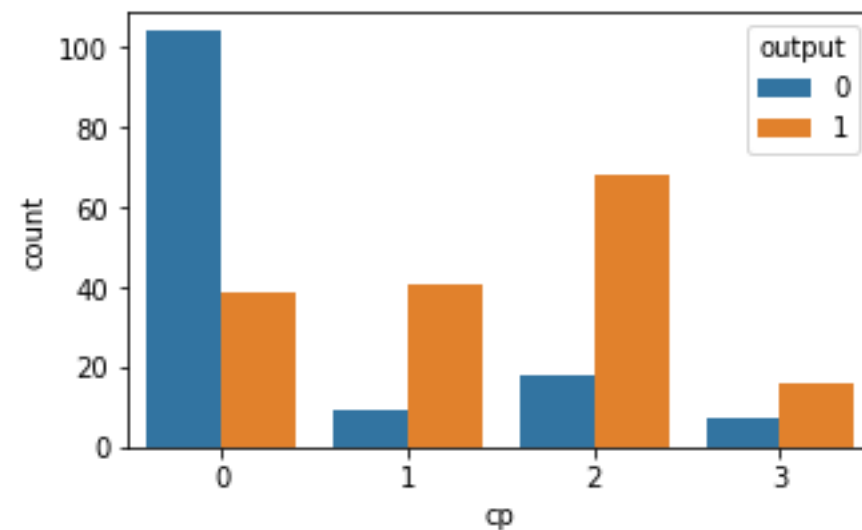
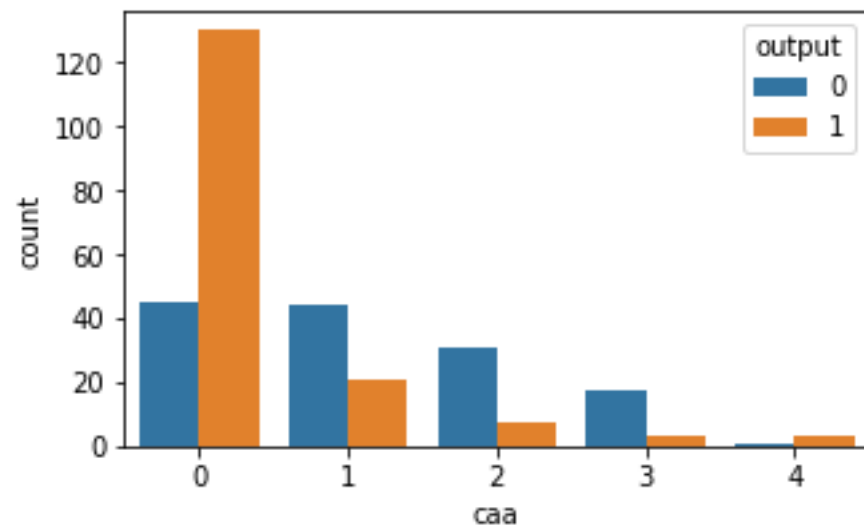
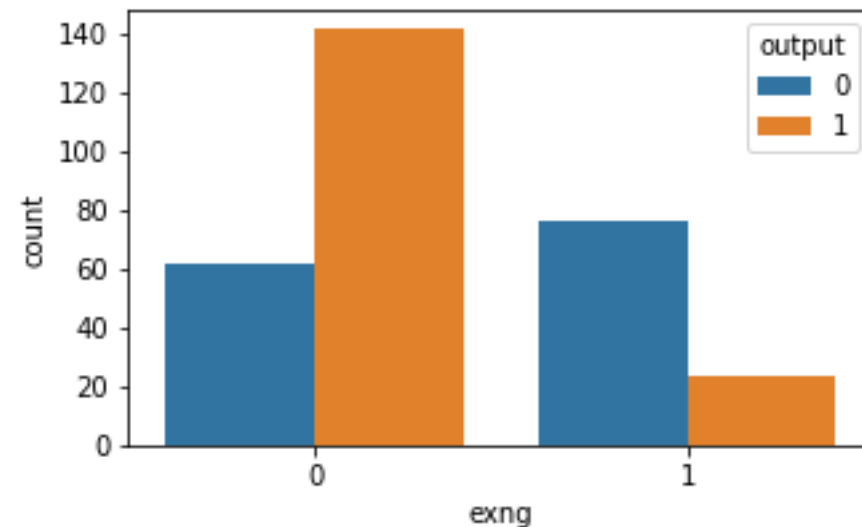
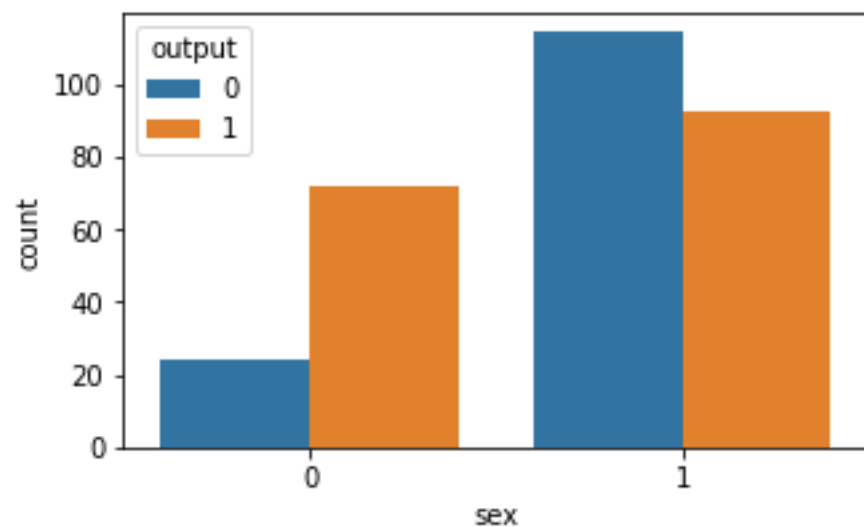
```
    print(
```

```
        sutun,":",
```

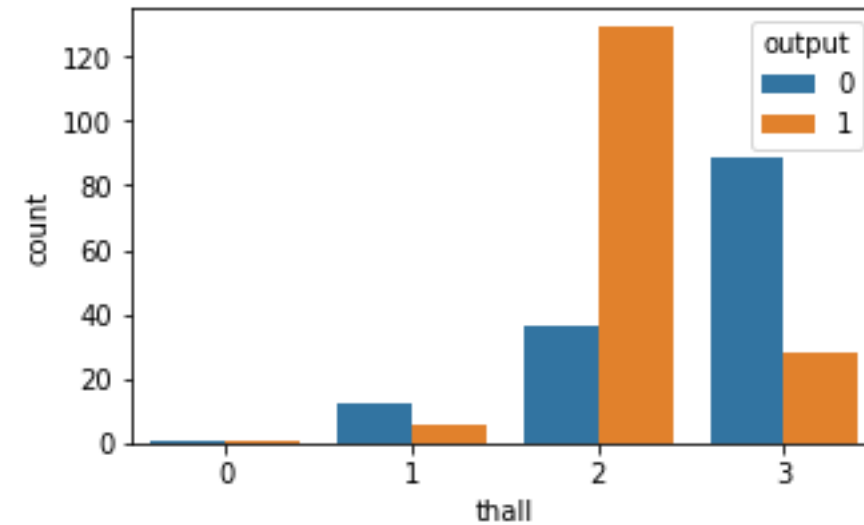
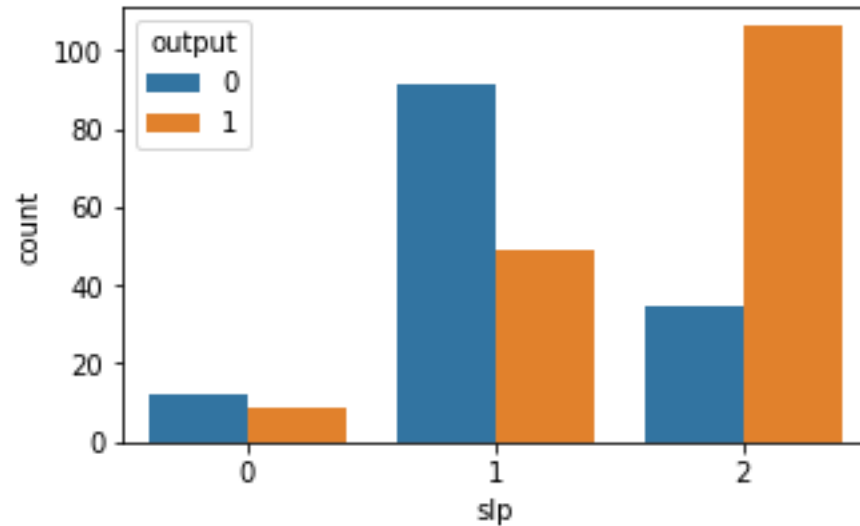
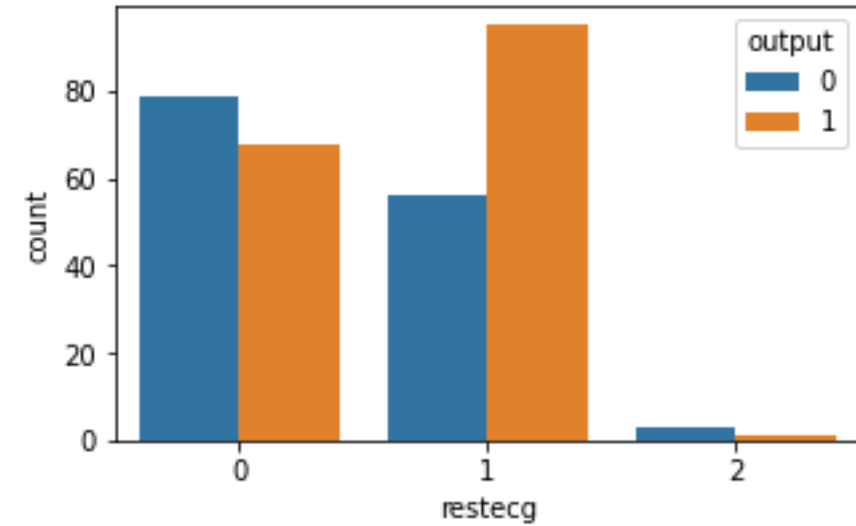
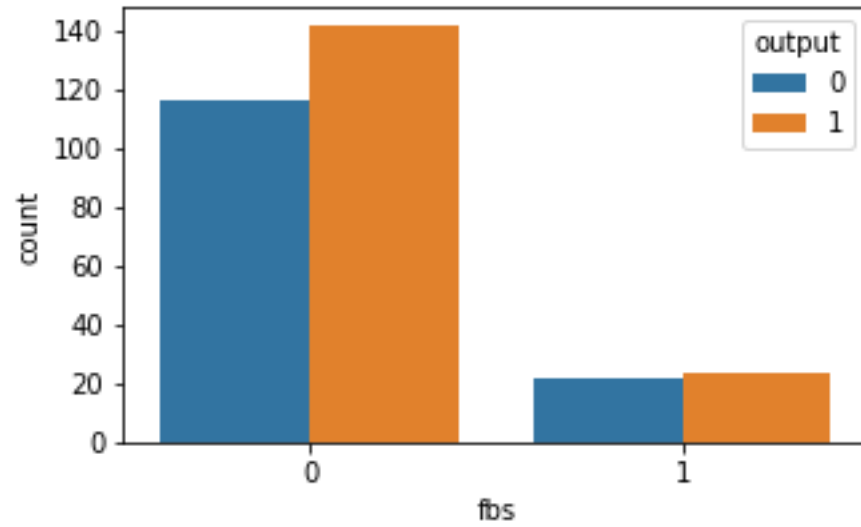
```
        len(data[sutun].unique())
```

```
)
```

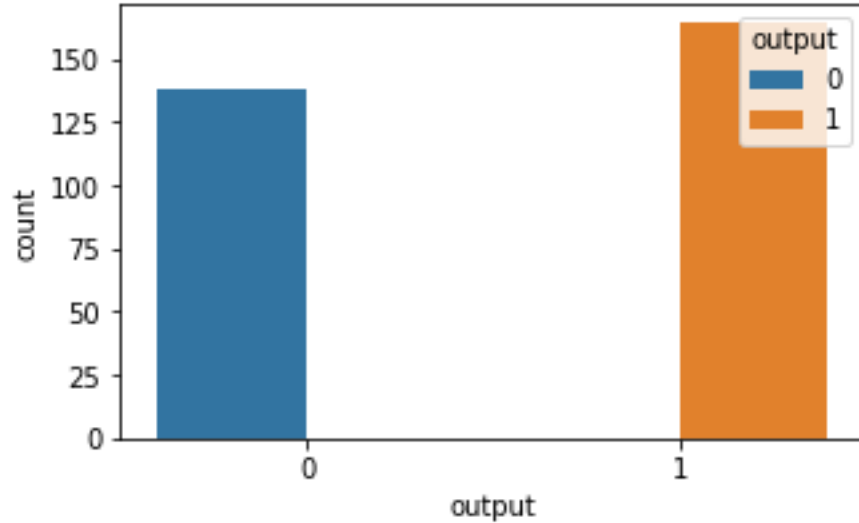
Kategorik olan verilerin grafikleri



Kategorik olan verilerin grafikleri



Kategorik olan verilerin grafikleri



```
### Kategorik olan verilerin grafikleri
```

```
kategorik_sutunlar = [
```

```
    'sex','exng','caa',
```

```
    'cp','fbs','restecg',
```

```
    'slp','thall','output'
```

```
]
```

```
for sutun in kategorik_sutunlar:
```

```
    plt.figure(figsize=(5,3))
```

```
    sns.countplot(
```

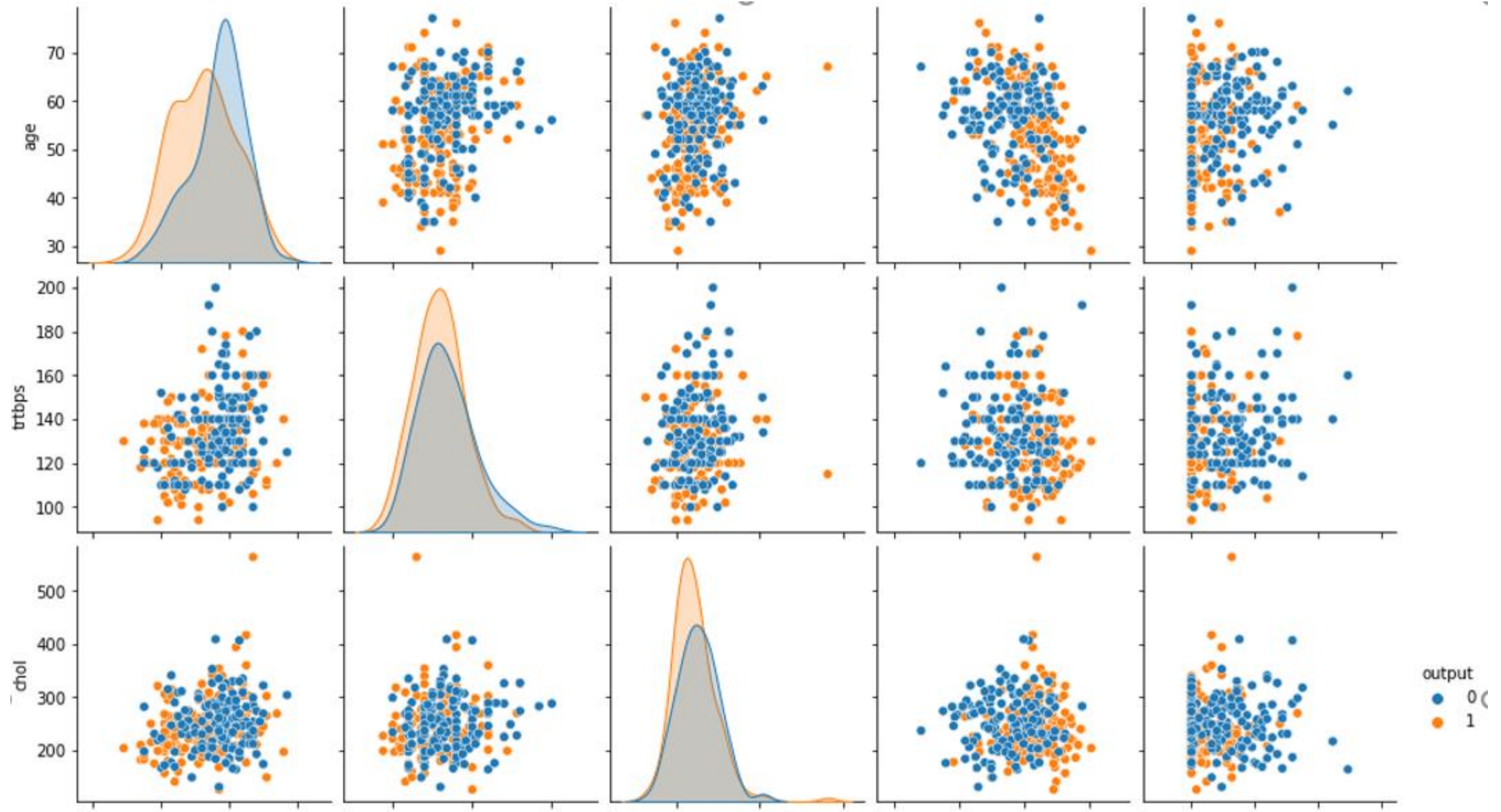
```
        x=sutun,
```

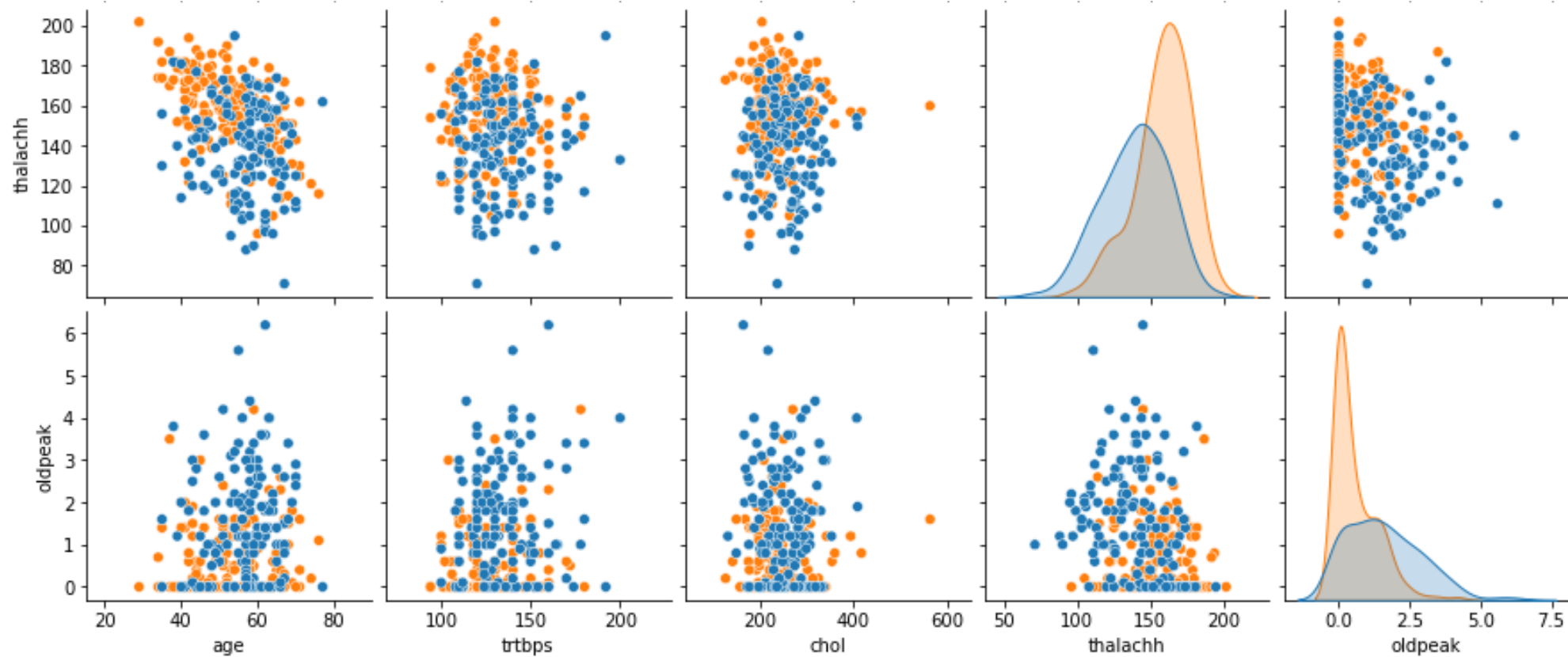
```
        data=data[kategorik_sutunlar],
```

```
        hue="output"
```

```
)
```

Devamlılık gösteren sütunlar





```
## pair plot
```

```
kategorik_olmayan =
```

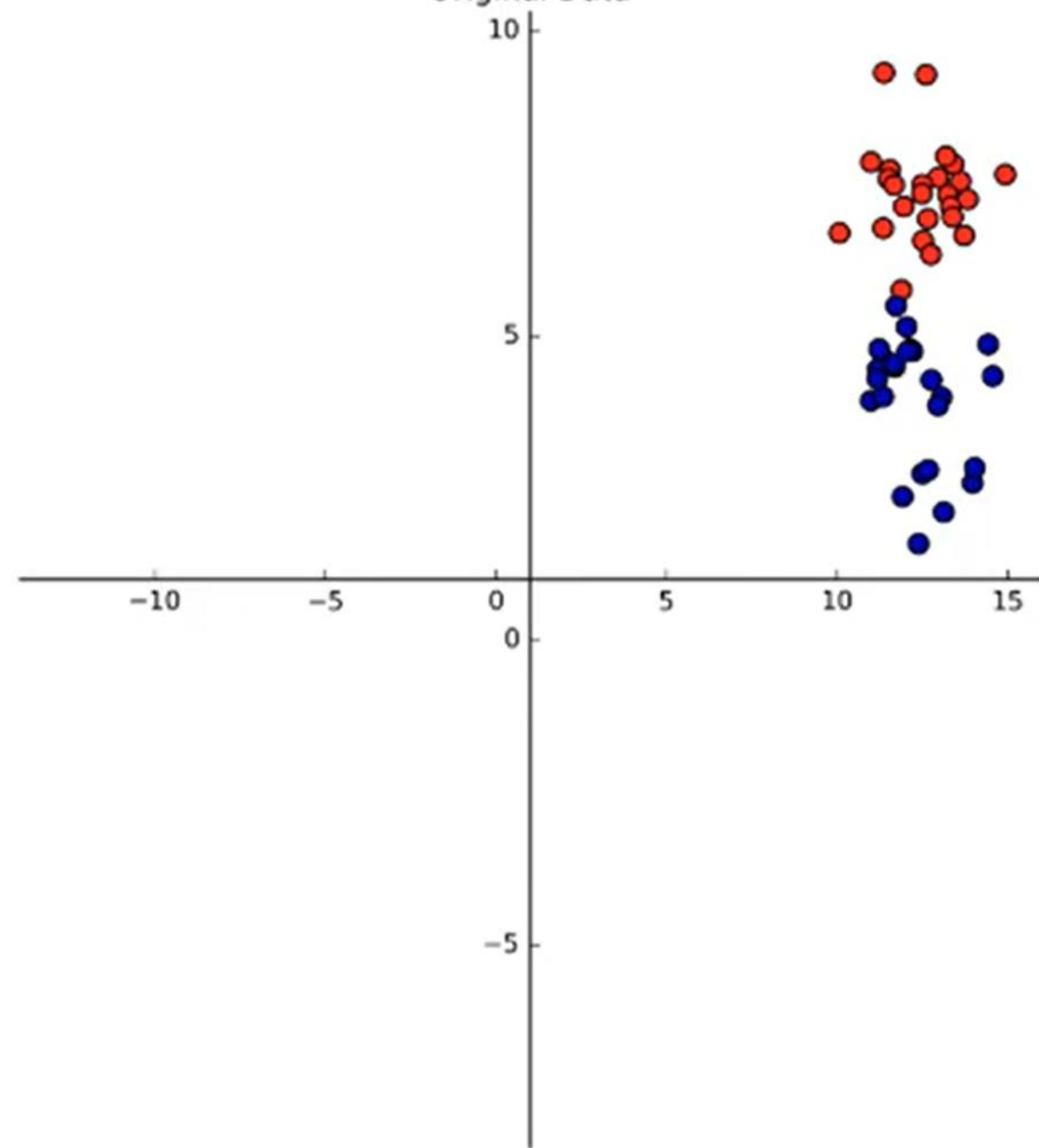
```
["age","trtbps","chol","thalachh","oldpeak","output"]
```

```
sns.pairplot(data[kategorik_olmayan], hue="output")
```

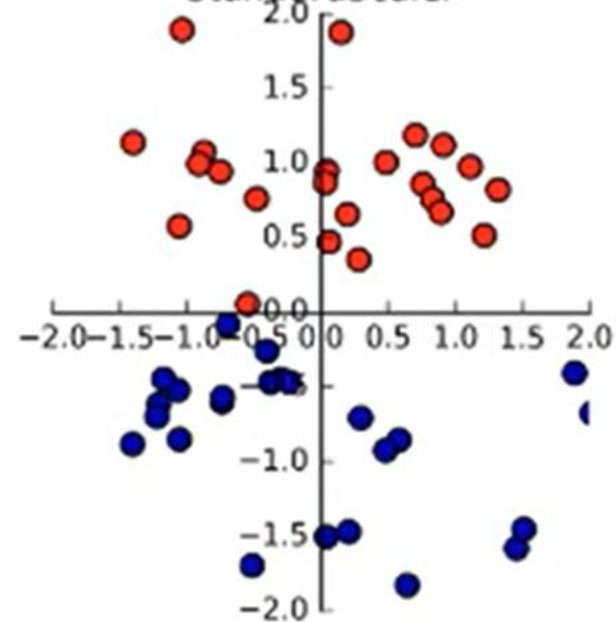
Veri ölçeklendirme

- Ölçekleme zorunlu değildir, yukarıdaki gibi ölçeklenmemiş bir data ile çok kötü bir sonuç alabiliriz.
- Ölçeklemenin temel amacı, daha büyük sayısal aralıkların etkilerinden kaçınmaktır
- Makine öğrenimi algoritmalarından önce verileri ölçeklendirmek daha iyi performans gösterecektir

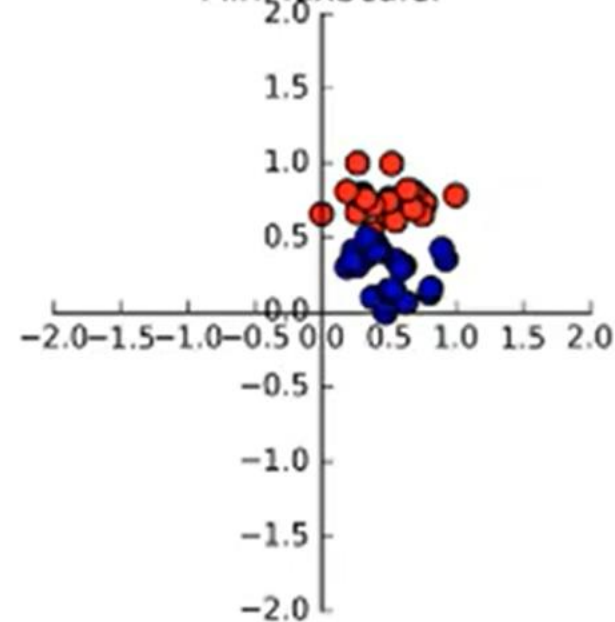
Original Data



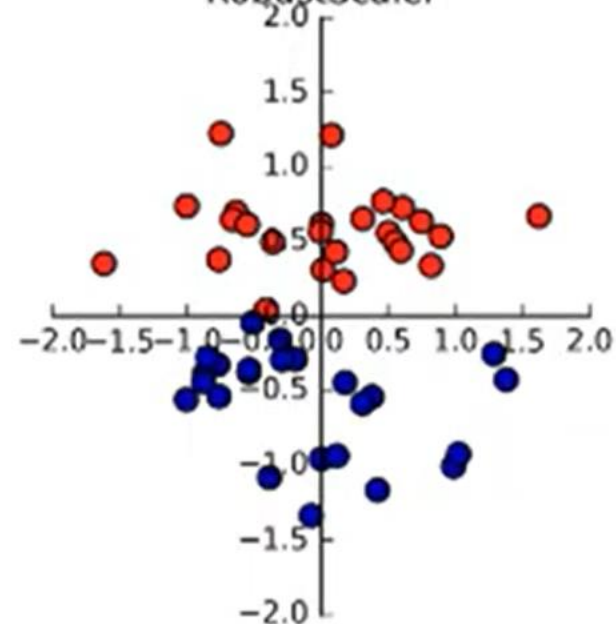
StandardScaler



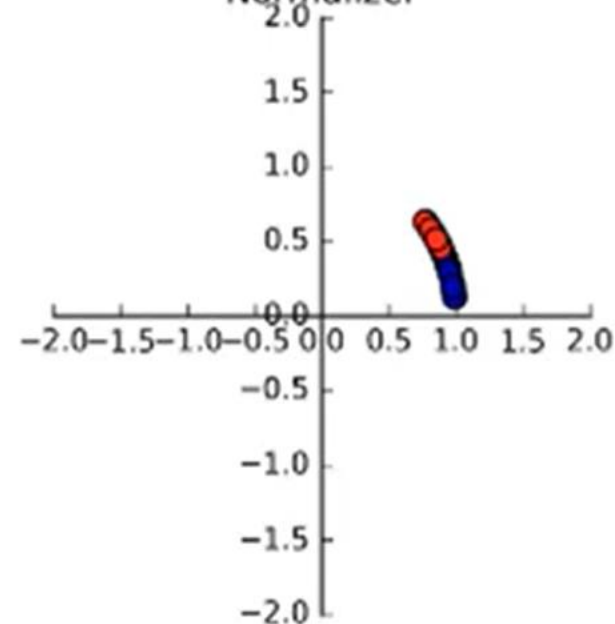
MinMaxScaler



RobustScaler



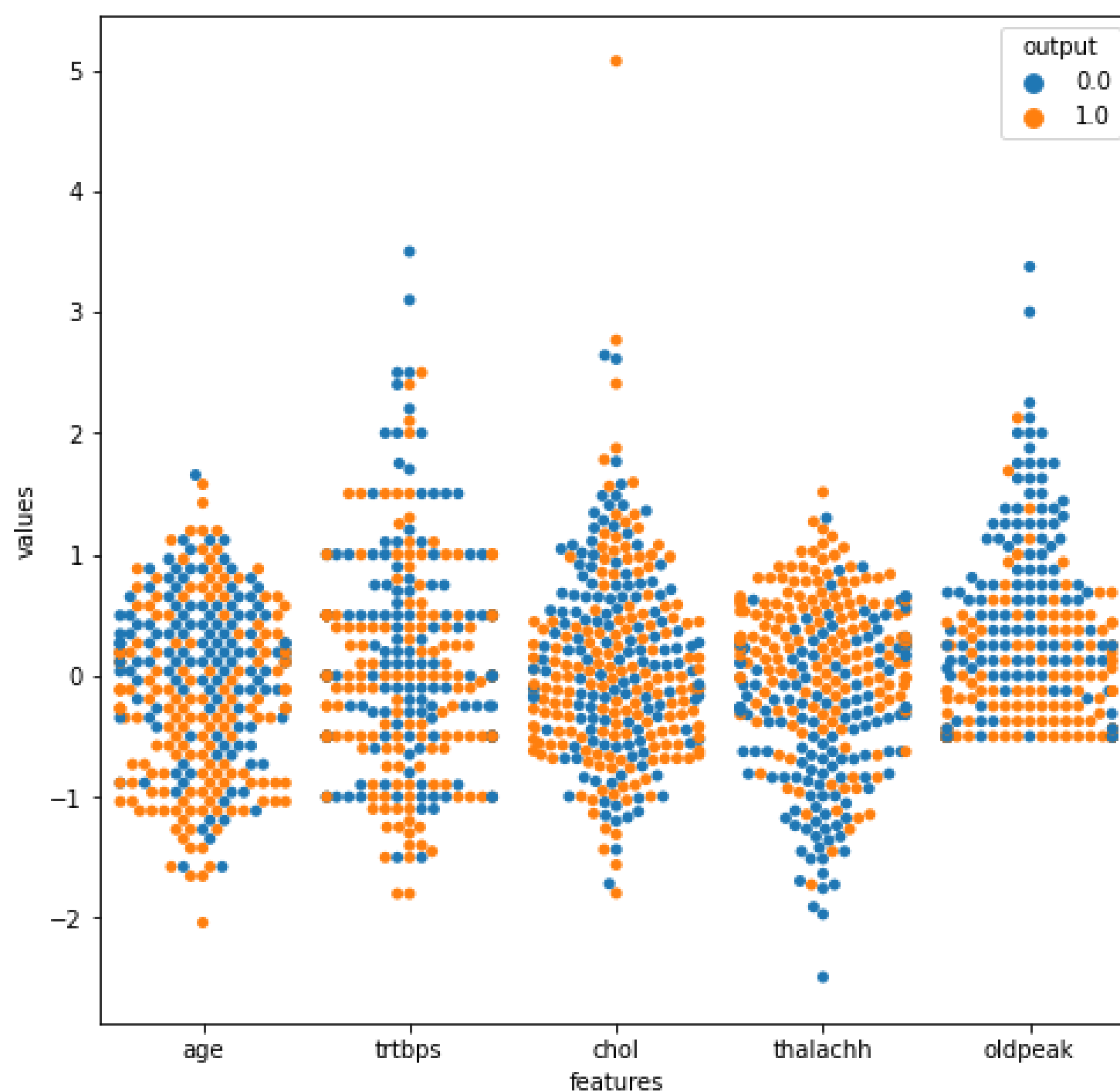
Normalizer



Robust Scaler

- RobustScaler, medyan tabanlı bir ölçekleme yöntemidir.
- RobustScaler formülü $(X_i - X_{median}) / IQR$ olduğundan aykırı değerlerden etkilenmez.

$$X_{new} = \frac{X - X_{median}}{IQR}$$



Swarm Plot

```
from sklearn.preprocessing import RobustScaler
```

```
kategorik_olmayan_data = data[kategorik_olmayan]
```

```
scaler = RobustScaler()
```

```
kategorik_olmayan_data =  
    scaler.fit_transform(  
kategorik_olmayan_data.drop(columns="output")  
)
```

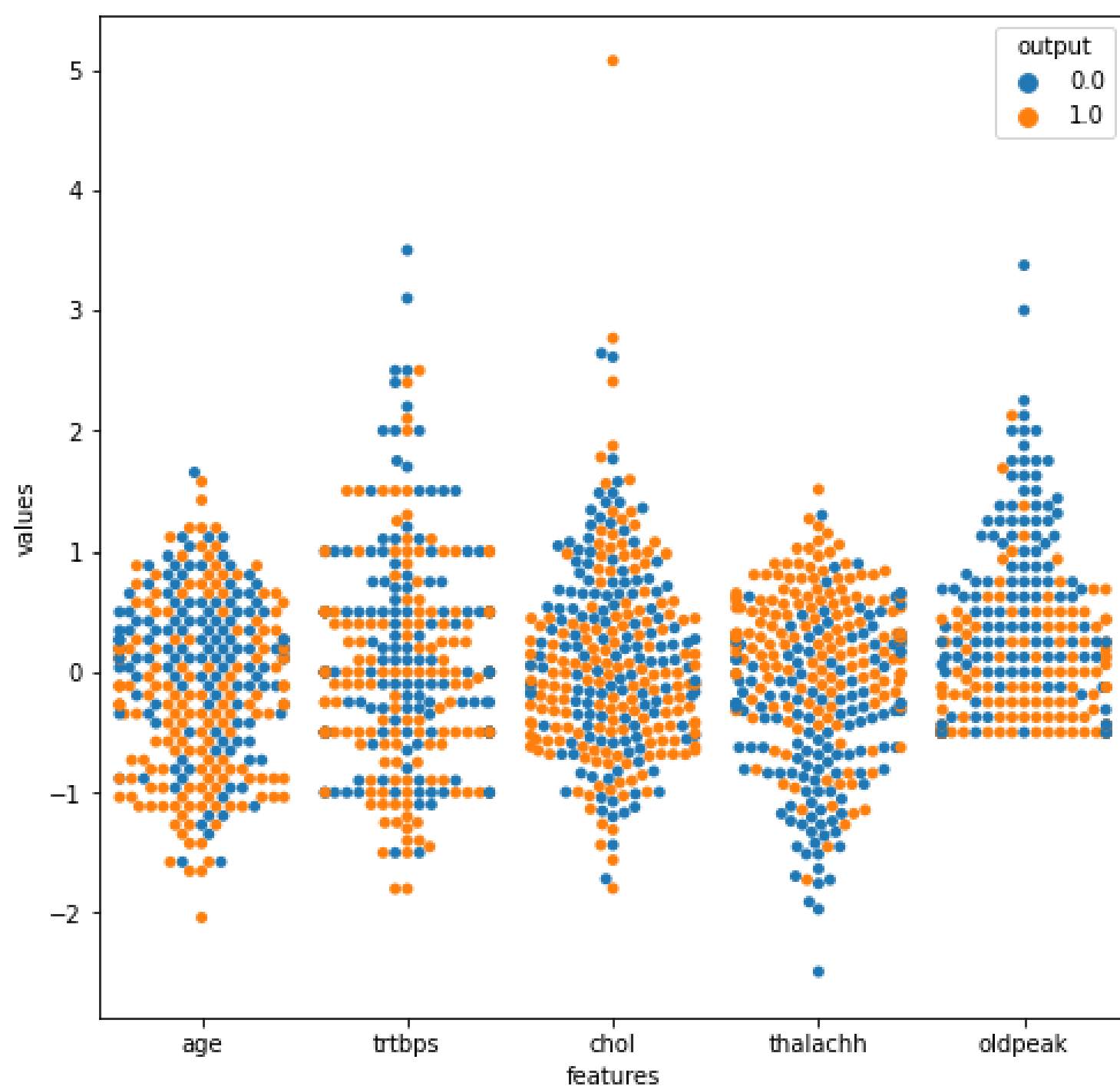
numpy array olduğu için dataframe çeviriyoruz

```
kategorik_olmayan_dataF =  
    pd.DataFrame(kategorik_olmayan_data,  
columns = kategorik_olmayan[:-1]  
)  
kategorik_olmayan_dataF.head()
```

```
In [5]: kategorik_olmayan_dataF.head()
```

```
Out[5]:
```

	age	trtbps	chol	thalachh	oldpeak	output
0	0.576923	0.75	-0.117647	-0.076336	0.9375	1.0
1	-1.423077	0.00	0.149020	1.053435	1.6875	1.0
2	-1.115385	0.00	-0.572549	0.595420	0.3750	1.0
3	0.038462	-0.50	-0.070588	0.778626	0.0000	1.0
4	0.115385	-0.50	1.780392	0.320611	-0.1250	1.0



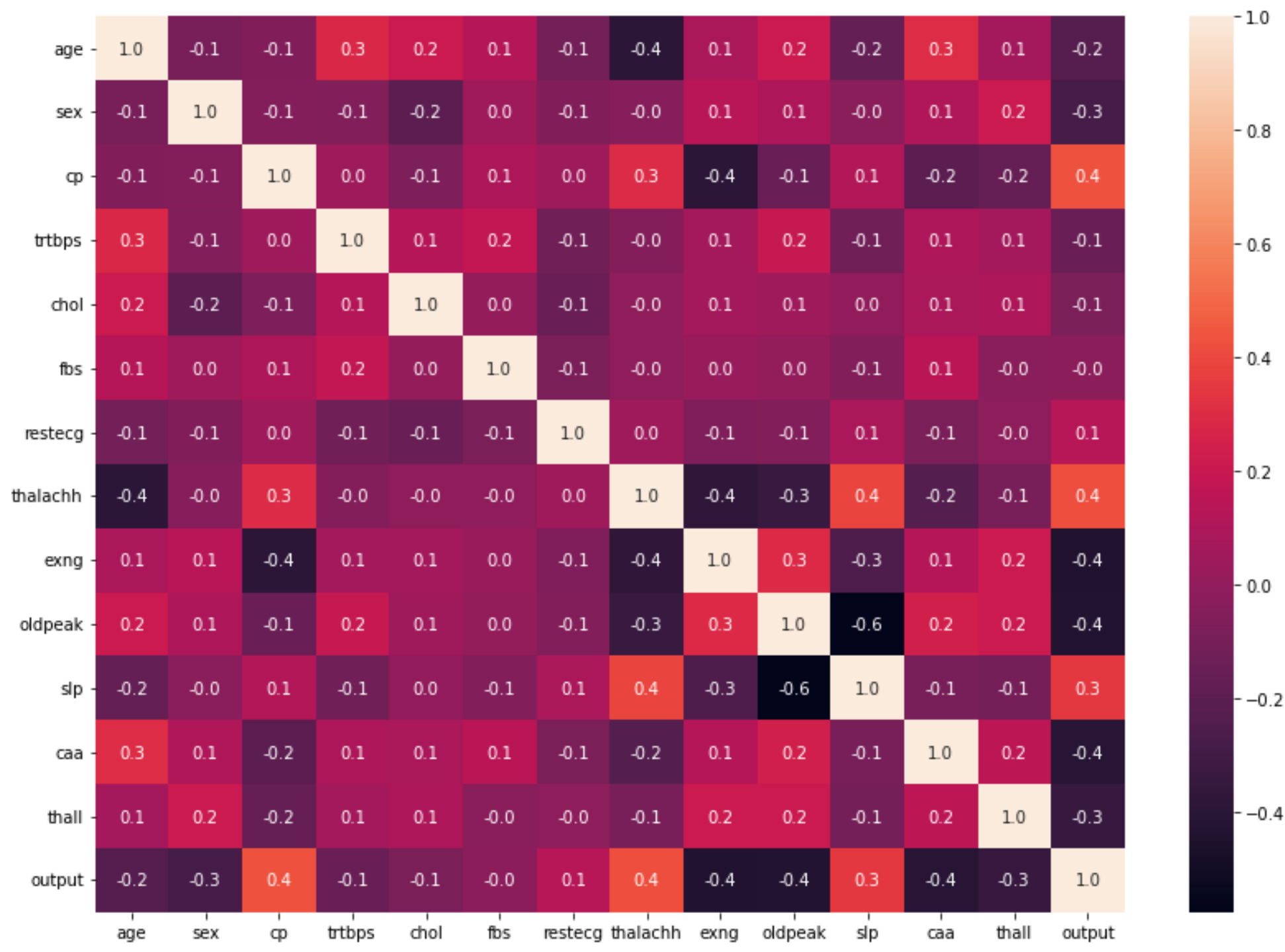
```
kategorik_olmayan_dataF =  
    pd.concat([kategorik_olmayan_dataF,  
data.loc[:, "output"]], axis = 1)
```

```
kategorik_olmayan_dataF.head()
```

```
kategorik_olmayan_dataF_melt = pd.melt(  
    kategorik_olmayan_dataF,  
    id_vars="output",  
    var_name="features",  
    value_name="values"  
)
```

```
kategorik_olmayan_dataF_melt.head()
```

```
plt.figure(figsize=(8,8))  
sns.swarmplot(  
    x="features",  
    y="values",  
    data=kategorik_olmayan_dataF_melt,  
    hue="output"  
)
```



CORRELATION MATRIX

```
# Heatmap
plt.figure(
    figsize=(14,10))
sns.heatmap(
    data.corr(),
    annot=True,
    fmt=".1f"
)
```

Dummy (Kukla) Variable

- **Integer Encoding:** Veride yer alan gruplara sayısal olarak bir değer ataması yapılır.
- {A = 1, B = 2, C = 3}
- **One-Hot Encoding:** 1 ve 0 olarak dönüştürülen gruplar vektör dizeleri halinde yeni değişkenler oluştururlar.
- A, B, C birer değişkene çevrilerek; {A : [1,0,0], B : [0,1,0], C : [0,0,1]}

kategorik_sutunlar - DataFrame

Index	sex	exnc	caa	co	slo	thall
85	0	0	0	2	1	3
86	1	0	1	2	2	3
87	1	0	0	1	2	3
88	0	0	0	2	1	2
89	0	0	0	0	1	2
90	1	0	2	2	2	2
91	1	1	0	0	2	3
92	1	0	4	2	2	2
93	0	1	1	1	2	2
94	0	0	0	1	1	2
95	1	1	0	0	2	3
96	0	0	0	0	1	2
97	1	0	3	0	2	3

df - DataFrame

Index	age	trtbns	chol	fbs	restecg	thalachh	oldpeak	output	sex 1	exng 1	caa 1	caa 2	caa 3	caa 4	cp 1	cp 2	cp 3	sln 1	sln 2	thall 1	thall 2
0	63	145	233	1	0	150	2.3	1	1	0	0	0	0	0	0	0	1	0	0	1	0
1	37	130	250	0	1	187	3.5	1	1	0	0	0	0	0	0	1	0	0	0	0	1
2	41	130	204	0	0	172	1.4	1	0	0	0	0	0	0	1	0	0	0	1	0	1
3	56	120	236	0	1	178	0.8	1	1	0	0	0	0	0	1	0	0	0	1	0	1
4	57	120	354	0	1	163	0.6	1	0	1	0	0	0	0	0	0	0	0	1	0	1
5	57	140	192	0	1	148	0.4	1	1	0	0	0	0	0	0	0	0	1	0	1	0
6	56	140	294	0	0	153	1.3	1	0	0	0	0	0	0	1	0	0	1	0	0	1
7	44	120	263	0	1	173	0	1	1	0	0	0	0	0	1	0	0	0	1	0	0
8	52	172	199	1	1	162	0.5	1	1	0	0	0	0	0	0	1	0	0	1	0	0
9	57	150	168	0	1	174	1.6	1	1	0	0	0	0	0	0	1	0	0	1	0	1
10	54	140	239	0	1	160	1.2	1	1	0	0	0	0	0	0	0	0	0	1	0	1
11	48	130	275	0	1	139	0.2	1	0	0	0	0	0	0	0	1	0	0	1	0	1
12	49	130	266	0	1	171	0.6	1	1	0	0	0	0	0	1	0	0	0	1	0	1
13	64	110	211	0	0	144	1.8	1	1	1	0	0	0	0	0	0	1	1	0	0	1
14	58	150	283	1	0	162	1	1	0	0	0	0	0	0	0	0	1	0	1	0	1
15	50	120	219	0	1	158	1.6	1	0	0	0	0	0	0	0	1	0	1	0	0	1
16	58	120	340	0	1	172	0	1	0	0	0	0	0	0	0	1	0	0	1	0	1
17	66	150	226	0	1	114	2.6	1	0	0	0	0	0	0	0	0	1	0	0	0	1
18	43	150	247	0	1	171	1.5	1	1	0	0	0	0	0	0	0	0	0	1	0	1
19	69	140	239	0	1	151	1.8	1	0	0	0	1	0	0	0	0	1	0	1	0	1
20	59	135	234	0	1	161	0.5	1	1	0	0	0	0	0	0	0	0	1	0	0	0

df = pd.get_dummies(data, columns = kategorik_sutunlar, drop_first = True)

Format

Resize

☒ Background color


☒ Column min/max

Save and Close

Close

```
kategorik_sutunlar = ['sex','exng','caa','cp','slp','thall']  
# Yaş, Maksimum kalp atış hızı, ST depresyonu  
kategorik_olmayan = ["age","thalachh","oldpeak"]  
# Dummy variable oluşturmak  
df = pd.get_dummies(data, columns = kategorik_sutunlar, drop_first = True)
```

```
x = df.drop(columns=['output',"chol","trtbps","fbs",'restecg'])  
y = df['output']  
x[kategorik_olmayan] = scaler.fit_transform(x[kategorik_olmayan])
```

 x - DataFrame

Index	age	thalachh	oldpeak	sex 1	exng 1	caa 1	caa 2	caa 3
0	0.576923	-0.0763359	0.9375	1	0	0	0	0

```
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=42)  
print("X_train :",X_train.shape)  
print("X_test :",X_test.shape)  
print("y_train :",y_train.shape)  
print("y_test :",y_test.shape)
```

```
X_train : (241, 17)  
X_test : (61, 17)  
y_train : (241,)  
y_test : (61,)
```

K-en yakın komşu algoritması

```
from sklearn.neighbors import KNeighborsClassifier
```

```
## Öklid uzaklığı -- euclidean
```

```
ken = KNeighborsClassifier(n_neighbors=5, metric="euclidean")  
ken.fit(X_train,y_train)
```

```
print("K-en yakın komşu algoritması: ",ken.score(X_test,y_test))
```

```
## Confusion Matrix
```

```
K-en yakın komşu algoritması:  0.8852459016393442  
[[26  3]  
 [ 4 28]]
```

```
ken_prediction = ken.predict(X_test)
```

```
print(confusion_matrix(y_test, ken_prediction))
```

```
confusion_heatmap(ken_prediction,"K-en yakın komşu algoritması")
```

Karar ağaçları

```
from sklearn.tree import DecisionTreeClassifier
```

```
tree = DecisionTreeClassifier()  
tree.fit(X_train,y_train)
```

```
print("Karar Ağaçları: ",tree.score(X_test,y_test))
```

```
## Confusion Matrix
```

```
decision_prediction = tree.predict(X_test)  
print(confusion_matrix(y_test, decision_prediction))  
confusion_heatmap(decision_prediction,"Karar Ağaçları")
```

```
Karar Ağaçları:  0.7704918032786885  
[[25  4]  
 [10 22]]
```

Logistic regresyon

```
from sklearn.linear_model import LogisticRegression
```

```
logreg = LogisticRegression()  
logreg.fit(X_train,y_train)
```

```
print("Logistic Regression: ",logreg.score(X_test,y_test))
```

```
## Confusion Matrix
```

```
logreg_prediction = logreg.predict(X_test)  
print(confusion_matrix(y_test, logreg_prediction))  
confusion_heatmap(logreg_prediction, "Logistic Regression")
```

```
Logistic Regression: 0.8852459016393442  
[[26  3]  
 [ 4 28]]
```


Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
```

```
nb = GaussianNB()  
nb.fit(X_train,y_train)  
print("Naive Bayes:",nb.score(X_test,y_test))
```

```
## Confusion Matrix  
nb_prediction = nb.predict(X_test)  
print(confusion_matrix(y_test, nb_prediction))  
confusion_heatmap(nb_prediction,"Naive Bayes")
```

```
Naive Bayes: 0.8524590163934426  
[[27  2]  
 [ 7 25]]  
|
```

Confusion heatmap Fonksiyonu

```
def confusion_heatmap(alg_predict, title):
```

```
    plt.figure()
```

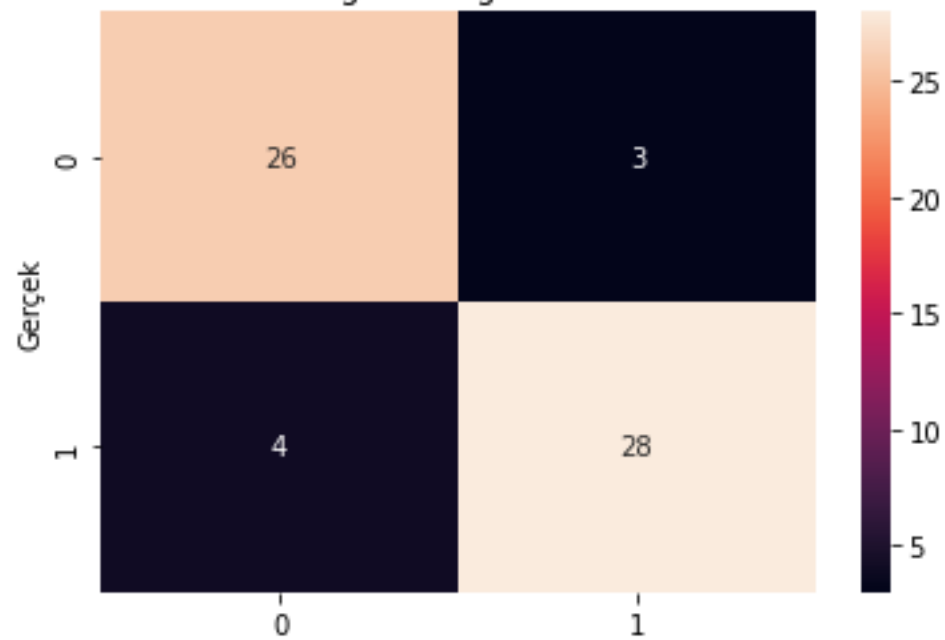
```
    sns.heatmap(confusion_matrix(y_test, alg_predict), annot=True)
```

```
    plt.xlabel("Tahmin")
```

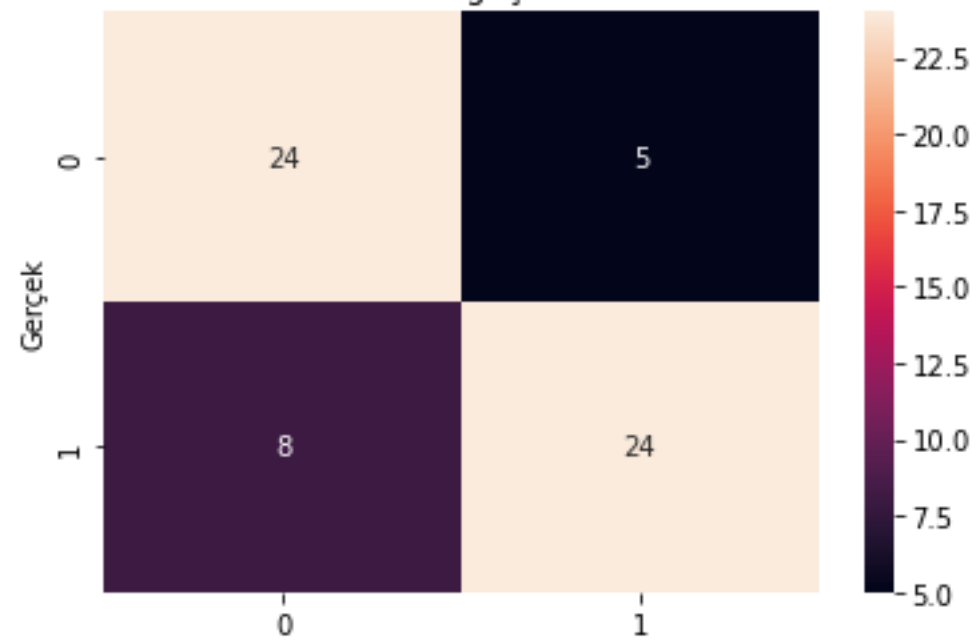
```
    plt.ylabel("Gerçek")
```

```
    plt.title(title)
```

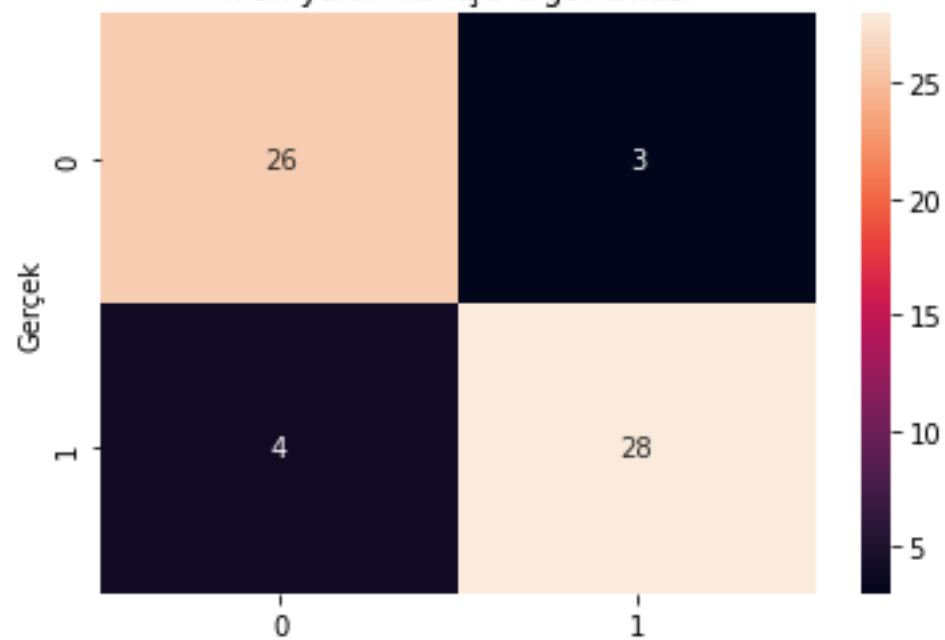
Logistic Regression



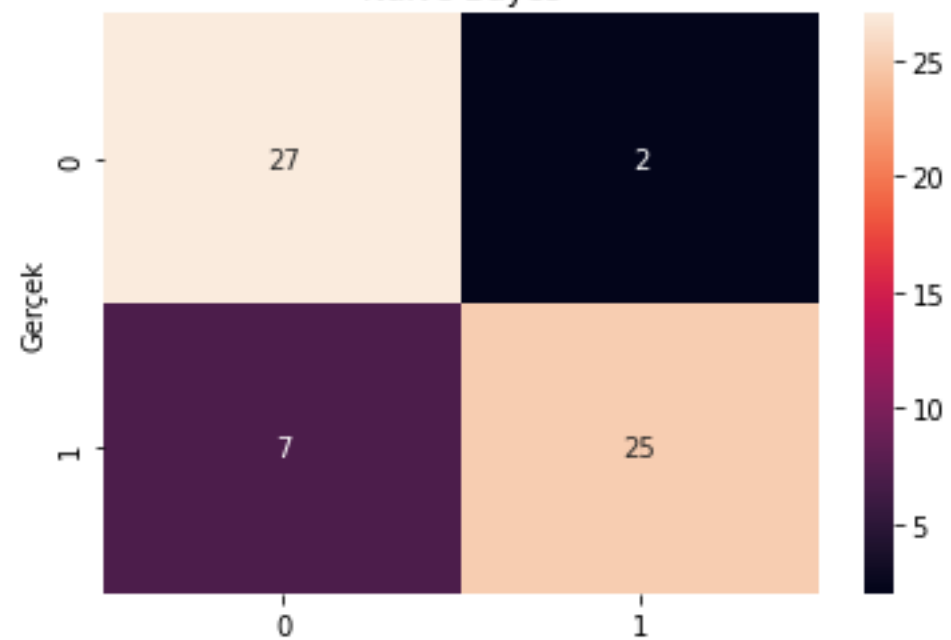
Karar Ağaçları



K-en yakın komşu algoritması



Naïve Bayes



Kaynak:

- [Heart Attack Analysis & Prediction Dataset | Kaggle](#)
- [scikit-learn: machine learning in Python — scikit-learn 1.2.0 documentation](#)
- [Scikit-Learn Kütüphanesi ile Data Ölçeklendirme | by İbrahim Halil Kaplan | Machine Learning Türkiye | Medium](#)
- [DENETİMSİZ ÖĞRENME | VERİ ÖLÇEKLEME | SCIKIT LEARN UYGULAMALARI | PYTHON MAKİNE ÖĞRENMESİ DERSLERİ – YouTube](#)
- [Veri Biliminde Kategorik Değişkenler, Dummy \(Kukla\) Variable ve Python Uygulaması | by Yiğit Şener | Medium](#)
- [K-NN \(K-NEAREST NEIGHBORS\) ALGORİTMASI İLE KALP KRİZİ TAHMİN VE ANALİZİ | by Fatma Çetin | Medium](#)