

---Room Occupancy detection data (IoT sensor)---

İzlenecek adım: Gerçek IoT verisi üzerinde gözetimli öğrenme algoritmalarını test edilecek. Amaç, verideki değişkenleri analiz ederek odada insan varlığını (occupancy) tespit etmek.

Üç adet veri seti mevcut:

- 1) Veriseti1: Kapılar kapalıyken alınmış ölçümler.
- 2) Veriseti2: Kapılar kapalıyken alınmış başka bir veri seti. Veriseti3: Kapılar açıkken alınmış ölçümler.
- 3) Hedef parametre: Occupancy. İnsan varlığı 1, insan yokluğu 0 değerlerine karşılık geliyor.

Kavramlar:

Temperature = Sıcaklık

Humidity = Nem

Light = Işık

CO2 = Oksijen

HumidityRatio = Nem Oranı

Occupancy = İnsan varlığı

```
# Gerekli kütüphaneler yüklendi.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import KFold, train_test_split, cross_val_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import plot_confusion_matrix
from sklearn.svm import SVC
%matplotlib inline

#Veriseti1 adlı text dosyası test1 adı ile okundu.
test1= pd.read_csv("Veriseti1.txt")
test1
```

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
1	2015-02-04 17:51:00	23.18	27.2720	426.0	721.250000	0.004793	1
2	2015-02-04 17:51:59	23.15	27.2675	429.5	714.000000	0.004783	1
3	2015-02-04 17:53:00	23.15	27.2450	426.0	713.500000	0.004779	1
4	2015-02-04 17:54:00	23.15	27.2000	426.0	708.250000	0.004772	1
5	2015-02-04 17:55:00	23.10	27.2000	426.0	704.500000	0.004757	1
...
8139	2015-02-10 09:29:00	21.05	36.0975	433.0	787.250000	0.005579	1
8140	2015-02-10 09:29:59	21.05	35.9950	433.0	789.500000	0.005563	1
8141	2015-02-10 09:30:59	21.10	36.0950	433.0	798.500000	0.005596	1
8142	2015-02-10 09:32:00	21.10	36.2600	433.0	820.333333	0.005621	1
8143	2015-02-10 09:33:00	21.10	36.2000	447.0	821.000000	0.005612	1

8143 rows × 7 columns

```
#Veriseti2 adlı text dosyası test2 adı ile okundu.
```

```
test2 = pd.read_csv("Veriseti2.txt")
test2
```

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
140	2015-02-02 14:19:00	23.700000	26.272000	585.200000	749.200000	0.004764	1
141	2015-02-02 14:19:59	23.718000	26.290000	578.400000	760.400000	0.004773	1
142	2015-02-02 14:21:00	23.730000	26.230000	572.666667	769.666667	0.004765	1
143	2015-02-02 14:22:00	23.722500	26.125000	493.750000	774.750000	0.004744	1
144	2015-02-02 14:23:00	23.754000	26.200000	488.600000	779.000000	0.004767	1
...
2800	2015-02-04 10:38:59	24.290000	25.700000	808.000000	1150.250000	0.004829	1
2801	2015-02-04 10:40:00	24.330000	25.736000	809.800000	1129.200000	0.004848	1
2802	2015-02-04 10:40:59	24.330000	25.700000	817.000000	1125.800000	0.004841	1
2803	2015-02-04 10:41:59	24.356667	25.700000	813.000000	1123.000000	0.004849	1
2804	2015-02-04 10:43:00	24.408333	25.681667	798.000000	1124.000000	0.004860	1

2665 rows × 7 columns

```
#Veriseti3 adlı text dosyası train adı ile okundu.
```

```
train = pd.read_csv("Veriseti3.txt")
train
```

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
1	2015-02-11 14:48:00	21.7600	31.133333	437.333333	1029.666667	0.005021	1
2	2015-02-11 14:49:00	21.7900	31.000000	437.333333	1000.000000	0.005009	1
3	2015-02-11 14:50:00	21.7675	31.122500	434.000000	1003.750000	0.005022	1
4	2015-02-11 14:51:00	21.7675	31.122500	439.000000	1009.500000	0.005022	1
5	2015-02-11 14:51:59	21.7900	31.133333	437.333333	1005.666667	0.005030	1
...
9748	2015-02-18 09:15:00	20.8150	27.717500	429.750000	1505.250000	0.004213	1
9749	2015-02-18 09:16:00	20.8650	27.745000	423.500000	1514.500000	0.004230	1
9750	2015-02-18 09:16:59	20.8900	27.745000	423.500000	1521.500000	0.004237	1
9751	2015-02-18 09:17:59	20.8900	28.022500	418.750000	1632.000000	0.004279	1
9752	2015-02-18 09:19:00	21.0000	28.100000	409.000000	1864.000000	0.004321	1

9752 rows × 7 columns

```
# test1 veri seti üzerindeki özelliklerin type bilgisi gösterildi.
```

```
test1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8143 entries, 1 to 8143
Data columns (total 7 columns):
#   Column          Non-Null Count  Dtype
---  -
0   date             8143 non-null   object
1   Temperature      8143 non-null   float64
2   Humidity         8143 non-null   float64
3   Light            8143 non-null   float64
4   CO2              8143 non-null   float64
5   HumidityRatio    8143 non-null   float64
6   Occupancy       8143 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 508.9+ KB
```

```
#test1 veri setine ait değerler içerisinde NaN olan değerlerin sayısını sıralayarak (sort komutu) ekrana yazdırıldı.
```

```
print(test1.isna().sum().sort_values())
```

```
date            0
Temperature     0
Humidity        0
Light           0
CO2             0
HumidityRatio   0
Occupancy       0
dtype: int64
```

```
# Uyarılar devre dışı bırakıldı.
import warnings
warnings.filterwarnings("ignore")

# test1, test2 ve train veri setleri üzerine Humidity ve
#HumidityRatio'nun toplamını gösteren "Nem + Nem oranı" adlı sütun eklendi.

test1['Nem + Nem oranı'] = test1['Humidity'] + test1['HumidityRatio']
test1 = test1.append(test1.iloc[:5])
test1.head(10)

test2['Nem + Nem oranı'] = test2['Humidity'] + test2['HumidityRatio']
test2 = test2.append(test2.iloc[:5])
test2.head(10)

train['Nem + Nem oranı'] = train['Humidity'] + train['HumidityRatio']
train = train.append(train.iloc[:5])
train.head(10)
```

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	Nem + Nem oranı
1	2015-02-11 14:48:00	21.7600	31.133333	437.333333	1029.666667	0.005021	1	31.138354
2	2015-02-11 14:49:00	21.7900	31.000000	437.333333	1000.000000	0.005009	1	31.005009
3	2015-02-11 14:50:00	21.7675	31.122500	434.000000	1003.750000	0.005022	1	31.127522
4	2015-02-11 14:51:00	21.7675	31.122500	439.000000	1009.500000	0.005022	1	31.127522
5	2015-02-11 14:51:59	21.7900	31.133333	437.333333	1005.666667	0.005030	1	31.138364
6	2015-02-11 14:53:00	21.7600	31.260000	437.333333	1014.333333	0.005042	1	31.265042
7	2015-02-11 14:54:00	21.7900	31.197500	434.000000	1018.500000	0.005041	1	31.202541
8	2015-02-11 14:55:00	21.7900	31.393333	437.333333	1018.666667	0.005073	1	31.398406
9	2015-02-11 14:55:59	21.7900	31.317500	434.000000	1022.000000	0.005060	1	31.322560
10	2015-02-11 14:57:00	21.7900	31.463333	437.333333	1027.333333	0.005084	1	31.468417

```
#date kısmı float değerler içermesi sebebiyle test1 veri seti içerisinden çıkarıldı.
test1.drop('date',inplace=True, axis=1)
test1.head(5)
```

	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	Nem + Nem oranı
1	23.18	27.2720	426.0	721.25	0.004793	1	27.276793
2	23.15	27.2675	429.5	714.00	0.004783	1	27.272283
3	23.15	27.2450	426.0	713.50	0.004779	1	27.249779
4	23.15	27.2000	426.0	708.25	0.004772	1	27.204772
5	23.10	27.2000	426.0	704.50	0.004757	1	27.204757

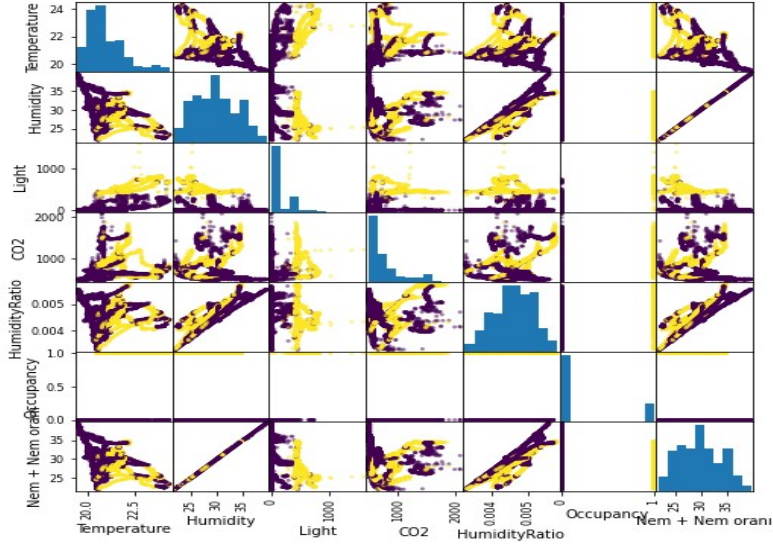
```
#date kısmı float değerler içermesi sebebiyle test2 veri seti içerisinden çıkarıldı.
test2.drop('date',inplace=True, axis=1)
test2.head(5)
```

	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	Nem + Nem oranı
140	23.7000	26.272	585.200000	749.200000	0.004764	1	26.276764
141	23.7180	26.290	578.400000	760.400000	0.004773	1	26.294773
142	23.7300	26.230	572.666667	769.666667	0.004765	1	26.234765
143	23.7225	26.125	493.750000	774.750000	0.004744	1	26.129744
144	23.7540	26.200	488.600000	779.000000	0.004767	1	26.204767

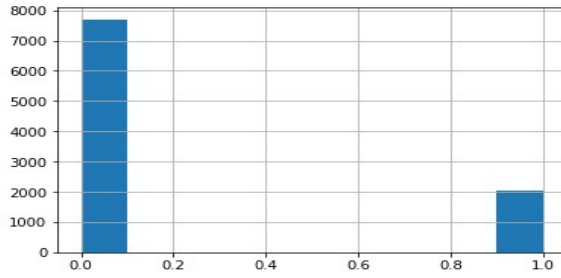
```
#date kısmı float değerler içermesi sebebiyle train veri seti içerisinden çıkarıldı.
train.drop('date',inplace=True, axis=1)
train.head(5)
```

	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	Nem + Nem oranı
1	21.7600	31.133333	437.333333	1029.666667	0.005021	1	31.138354
2	21.7900	31.000000	437.333333	1000.000000	0.005009	1	31.005009
3	21.7675	31.122500	434.000000	1003.750000	0.005022	1	31.127522
4	21.7675	31.122500	439.000000	1009.500000	0.005022	1	31.127522
5	21.7900	31.133333	437.333333	1005.666667	0.005030	1	31.138364


```
# Saçılım matrisi Occupancy parametresine göre çizildi.
pd.plotting.scatter_matrix(train, c=train['Occupancy'], figsize=[8, 8])
plt.show()
```



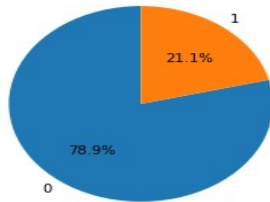
```
# train datasetinin Occupancy parametresine göre histogram grafiğinin dağılımı.
train["Occupancy"].hist();
```



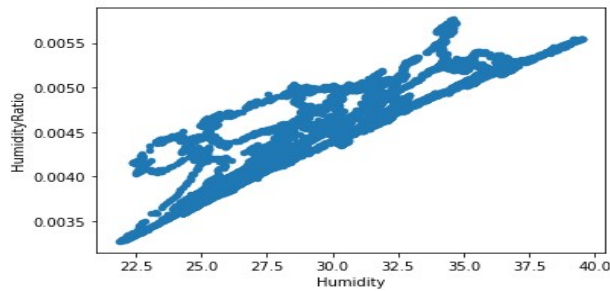
```
# train veriseti içindeki oranlar dilimsel olarak gösterildi.
labels=['0', '1']
Pie_chart=train.Occupancy.value_counts()
plt.pie(Pie_chart, labels=labels, startangle=90, autopct='%1.1f%%')
plt.title("Occupancy parametresinin dilim grafiği üzerinde dağılımı")
```

Text(0.5, 1.0, 'Occupancy parametresinin dilim grafiği üzerinde dağılımı')

Occupancy parametresinin dilim grafiği üzerinde dağılımı

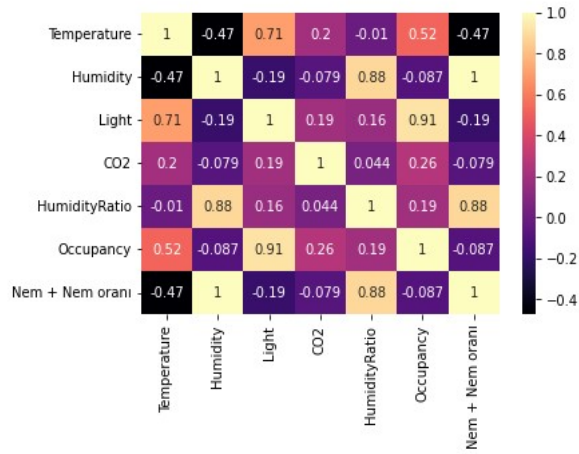


```
# x ve y ye göre scatter dağılımı çizdirildi.
train.plot(x="Humidity", y="HumidityRatio", kind="scatter");
```



```
# Isı Haritası grafiği gösterildi.
corr = train.corr(method='pearson')
sns.heatmap(corr, cmap="magma", annot=True)
```

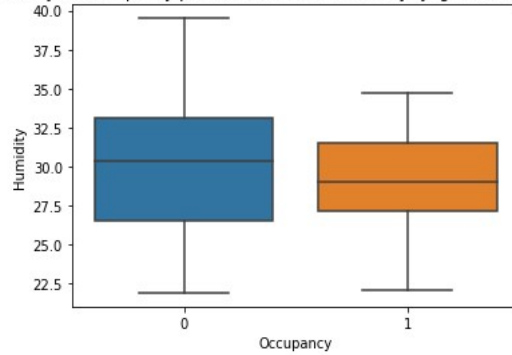
<AxesSubplot:>



```
# Kutu grafiği.
sns.boxplot(x="Occupancy",y='Humidity',data=train)
plt.title("Humidity ve Occupancy parametreleri arasındaki ilişkiyi gösteren kutu grafiği")
```

Text(0.5, 1.0, 'Humidity ve Occupancy parametreleri arasındaki ilişkiyi gösteren kutu grafiği')

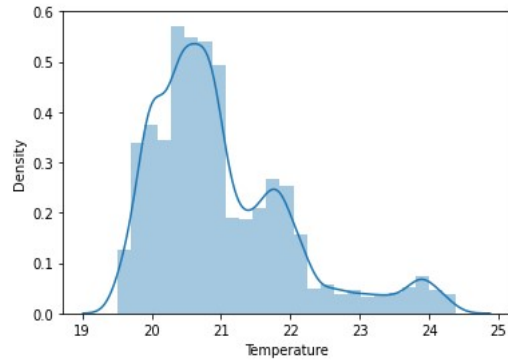
Humidity ve Occupancy parametreleri arasındaki ilişkiyi gösteren kutu grafiği



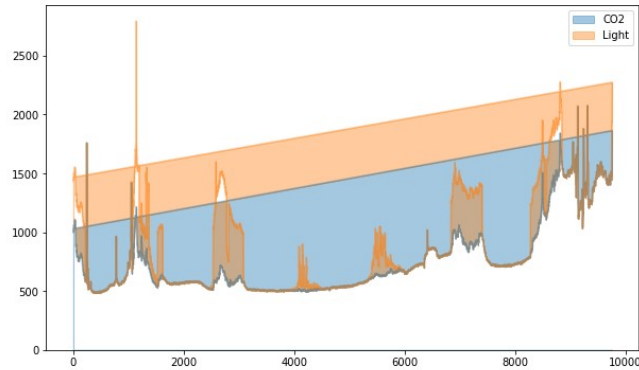
```
# Uyarılar devre dışı bırakıldı.
import warnings
warnings.filterwarnings("ignore")

#Temperature parametresinin dağılımını gösteren grafik çizdirildi.
sns.distplot(train['Temperature'],bins=25,kde=True)
```

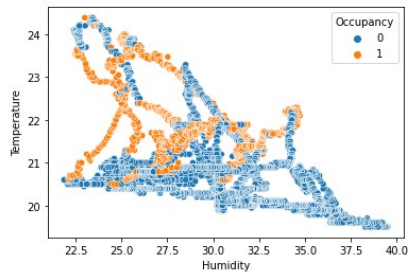
<AxesSubplot:xlabel='Temperature', ylabel='Density'>



```
# Light ve CO2 arasındaki ilişkiyi gösteren alan grafiği çizdirildi.
train.plot.area(y=['CO2','Light'],alpha=0.4,figsize=(10, 6));
```



```
sns.scatterplot(x = "Humidity", y = "Temperature", hue = "Occupancy", data = train)
<AxesSubplot:xlabel='Humidity', ylabel='Temperature'>
```



Random forest:

```
#### Random Forest

from sklearn.ensemble import RandomForestClassifier

X_test1 = test1.drop('Occupancy', axis=1)
y_test1 = test1['Occupancy']

X_test2 = test2.drop('Occupancy', axis=1)
y_test2 = test2['Occupancy']

X = train.drop('Occupancy',axis=1)
# Occupancy etiketi y değişkenine atandı.
y = train['Occupancy']

# Veri test ve train olarak ayrıldı.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1)

# k - fold 5 olarak ayarlandı.
n_folds = 5
# Random Forest için hiper-parametre seçimi yapıldı.
rfc = RandomForestClassifier(max_depth=10, n_estimators=15,
                             max_features=0.35, random_state=42,oob_score=True)

from sklearn.model_selection import GridSearchCV
hyper_params={'min_samples_leaf':[10],
              'min_samples_split':[15]}
}
```

```
# Uyarılar devre dışı bırakıldı.
import os
import warnings
warnings.filterwarnings('ignore')

# rfc modeli fit edildi.
rfc.fit(X_train,y_train)
# tahmin yapıldı.
y_pred = rfc.predict(X_test)

accuracy_score(y_test, y_pred)
#En son kısımda modellerin karşılaştırılması için accuracy_rfc ve acc_rfc adlı kod çıktıları oluşturuldu.
accuracy_rfc=round(accuracy_score(y_test,y_pred)* 100, 2)
acc_rfc = round(rfc.score(X_train, y_train) * 100, 2)

# Confusion matrisi sayısal olarak ekrana yazdırıldı.
cm = confusion_matrix(y_test, y_pred)
print('Random Forest için Confusion matrisi \n',cm)

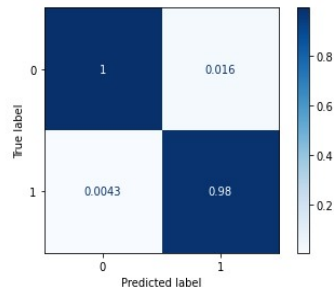
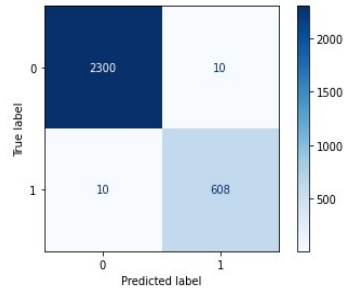
Random Forest için Confusion matrisi
[[2300  10]
 [ 10 608]]
```

```
# Sınıflandırma raporu ekrana yazdırıldı.
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2310
1	0.98	0.98	0.98	618
accuracy			0.99	2928
macro avg	0.99	0.99	0.99	2928
weighted avg	0.99	0.99	0.99	2928

```
# Confusion matrisi sayısal ve oransal olarak ekrana yazdırıldı.
plot_confusion_matrix(rfc,X_test, y_test,cmap= 'Blues')
plot_confusion_matrix(rfc,X_test, y_test,cmap= 'Blues',normalize= 'pred')
```

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f16ba4a650>



Logistic Regression :

```
#### Logistic Regression
```

```
from sklearn.linear_model import LogisticRegression

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

# tahmin yapılması
y_pred = logreg.predict(X_test)

X = train.drop('Occupancy',axis=1)

# Occupancy etiketi y değişkenine atandı.
y = train['Occupancy']

# Veri test ve train olarak ayrıldı.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1)

y_pred = logreg.predict(X_test)
accuracy_score(y_test, y_pred)
accuracy_log=round(accuracy_score(y_test,y_pred)* 100, 2)
acc_log = round(logreg.score(X_train, y_train) * 100, 2)

cm = confusion_matrix(y_test, y_pred)
print('Logistik regresyon için Confusion matrisi\n',cm)

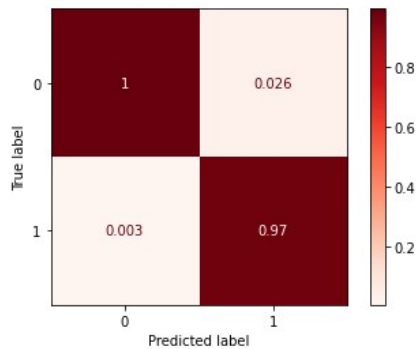
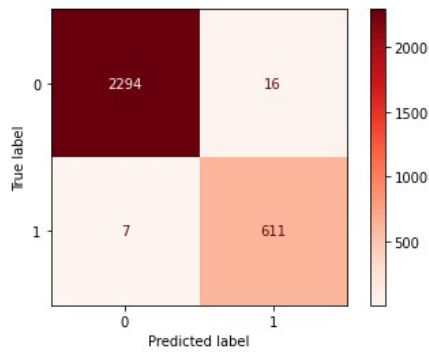
Logistik regresyon için Confusion matrisi
[[2294  16]
 [   7 611]]
```

```
# Sınıflandırma raporu ekrana yazdırıldı.
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	1.00	0.99	1.00	2310
1	0.97	0.99	0.98	618
accuracy			0.99	2928
macro avg	0.99	0.99	0.99	2928
weighted avg	0.99	0.99	0.99	2928

```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(logreg,X_test, y_test,cmap= 'Reds')
plot_confusion_matrix(logreg,X_test, y_test,cmap= 'Reds',normalize= 'pred')

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f16b008a60>
```



Decision Tree:

```
#### Decision Tree (KARAR AĞAÇLARI)

#Model Seçimine göre kütüphane belirlendi.
from sklearn.tree import DecisionTreeClassifier

X = train.drop('Occupancy',axis=1)

# Occupancy etiketi y değişkenine atandı.
y = train['Occupancy']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2, random_state=42)

dt = DecisionTreeClassifier(max_depth=2).fit(X=X_train, y=y_train)
y_pred = dt.predict(X_test)

accuracy_score(y_test, y_pred)
accuracy_dt=round(accuracy_score(y_test,y_pred)* 100, 2)
acc_dt = round(dt.score(X_train, y_train) * 100, 2)
cm = confusion_matrix(y_test, y_pred)
print('Karar ağacı için Confusion matrisi \n',cm)
```

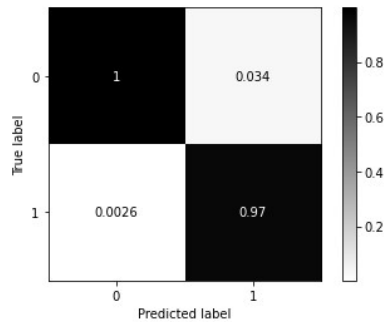
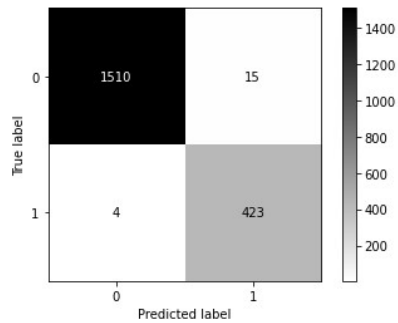
```
Karar ağacı için Confusion matrisi
[[1510  15]
 [  4 423]]
```

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	1.00	0.99	0.99	1525
1	0.97	0.99	0.98	427
accuracy			0.99	1952
macro avg	0.98	0.99	0.99	1952
weighted avg	0.99	0.99	0.99	1952


```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(dt,X_test, y_test,cmap= 'gist_yarg')
plot_confusion_matrix(dt,X_test, y_test,cmap= 'gist_yarg',normalize= 'pred')

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f16be1aec0>
```



Destek Vektör Makinesi:

```
#### Destek Vektör Makinesi (Support vector machine-SVM))

X = train.drop('Occupancy',axis=1)

# Occupancy etiketi y değişkenine atandı.
y = train['Occupancy']

# Veri test ve train olarak ayrıldı.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1)

svm=SVC(kernel='linear').fit(X_train, y_train)
#print(Svm.score(X_train, y_train))

y_pred=svm.predict(X_test)
#print(Svm.score(X_test,y_test))

accuracy_score(y_test, y_pred)
accuracy_SVM=round(accuracy_score(y_test,y_pred)* 100, 2)
acc_SVM = round(svm.score(X_train, y_train) * 100, 2)
cm = confusion_matrix(y_test, y_pred)
print('Destek Vektör Makinesi için Confusion matrisi\n',cm)
```

Destek Vektör Makinesi için Confusion matrisi

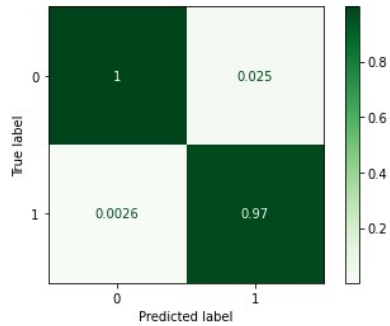
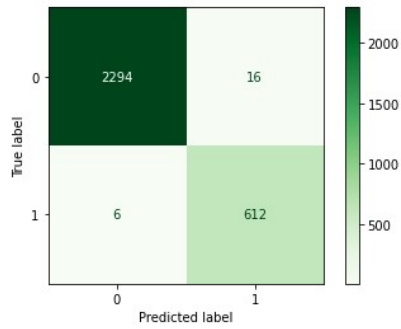
```
[[2294  16]
 [   6 612]]
```

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	1.00	0.99	1.00	2310
1	0.97	0.99	0.98	618
accuracy			0.99	2928
macro avg	0.99	0.99	0.99	2928
weighted avg	0.99	0.99	0.99	2928

```
plot_confusion_matrix(svm,X_test, y_test,cmap= 'Greens')
plot_confusion_matrix(svm,X_test, y_test,cmap= 'Greens',normalize= 'pred')
```

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f16d204c10>



K Nearest Neighbor (KNN):

```
##### K Nearest Neighbor(KNN)

from sklearn.neighbors import KNeighborsClassifier

X = train.drop('Occupancy',axis=1)
# Occupancy etiketi y değişkenine atandı.
y = train['Occupancy']

# Veri test ve train olarak ayrıldı.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1)

knn = KNeighborsClassifier(n_neighbors = 5)

knn.fit(X_train, y_train)
print(knn.score(X_train, y_train))
y_pred=knn.predict(X_test)
print(knn.score(X_test,y_test))

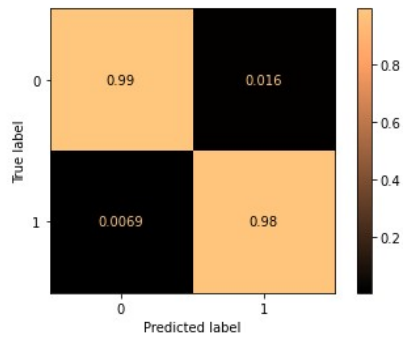
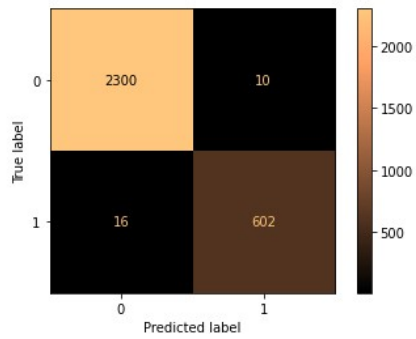
accuracy_score(y_test, y_pred)
accuracy_knn=round(accuracy_score(y_test,y_pred)* 100, 2)
acc_knn = round(knn.score(X_train, y_train) * 100, 2)
cm = confusion_matrix(y_test, y_pred)
print(' K-En Yakın Komşu Algoritması için Confusion matrisi\n',cm)
```

```
0.9944354956801874
0.9911202185792349
K-En Yakın Komşu Algoritması için Confusion matrisi
[[2300  10]
 [ 16 602]]
```

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.99	1.00	0.99	2310
1	0.98	0.97	0.98	618
accuracy			0.99	2928
macro avg	0.99	0.98	0.99	2928
weighted avg	0.99	0.99	0.99	2928

```
plot_confusion_matrix(knn,X_test, y_test,cmap= 'copper')
plot_confusion_matrix(knn,X_test, y_test,cmap= 'copper',normalize= 'pred')
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f16ba36680>
```



Gaussian Naive Bayes:

```
##### Gaussian Naive Bayes

from sklearn.naive_bayes import GaussianNB

# X değişkenine etiket hariç tüm kolonları atayalım.
X = train.drop('Occupancy',axis=1)

# Occupancy etiketi y değişkenine atandı.
y = train['Occupancy']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1)

NB = GaussianNB()
NB.fit(X_train, y_train)

y_pred = NB.predict(X_test)
accuracy_nb=round(accuracy_score(y_test,y_pred)* 100, 2)
acc_nb = round(NB.score(X_train, y_train) * 100, 2)
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test,y_pred)

print('Naive Bayes için Confusion matrisi\n',cm)
```

```
Naive Bayes için Confusion matrisi
[[2208  102]
 [   2  616]]
```

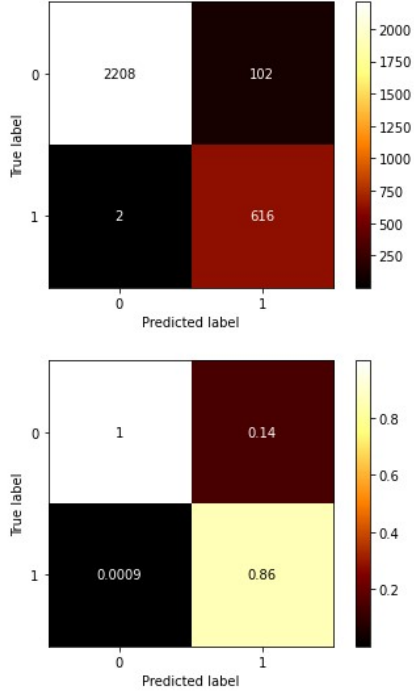
```
# Sınıflandırma raporu ekrana yazdırıldı.
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	1.00	0.96	0.98	2310
1	0.86	1.00	0.92	618
accuracy			0.96	2928
macro avg	0.93	0.98	0.95	2928
weighted avg	0.97	0.96	0.97	2928

```

plot_confusion_matrix(NB,X_test, y_test,cmap= 'afmhot')
plot_confusion_matrix(NB,X_test, y_test,cmap= 'afmhot',normalize= 'pred')
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f16bb33460>

```



: # Altı farklı gözetimli öğrenme modelinin Accuracy_score üzerinden karşılaştırılması yapıldı.

```

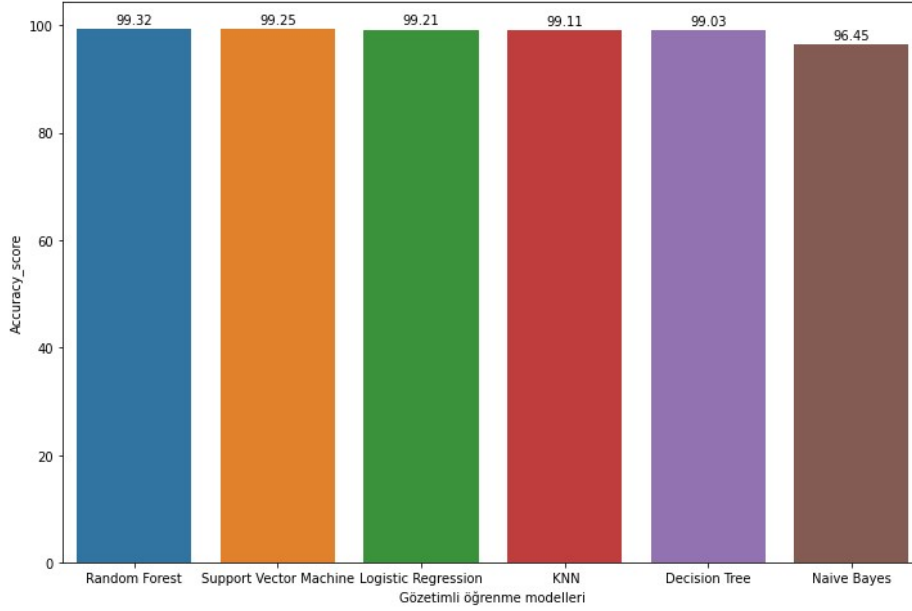
sonuc = pd.DataFrame({
    'Gözetimli öğrenme modelleri': ['Random Forest',
    'Logistic Regression',
    'Decision Tree',
    'Support Vector Machine',
    ' KNN',
    'Naive Bayes'],
    'Score': [acc_rfc,
    acc_log,
    acc_dt,
    acc_SVM,
    acc_knn,
    acc_nb],
    "Accuracy_score": [accuracy_rfc,
    accuracy_log,
    accuracy_dt,
    accuracy_SVM,
    accuracy_knn,
    accuracy_nb]})
sonuc_df = sonuc.sort_values(by='Accuracy_score', ascending=False)
sonuc_df = sonuc_df.reset_index(drop=True)
sonuc_df.head(6)

```

	Gözetimli öğrenme modelleri	Score	Accuracy_score
0	Random Forest	99.85	99.32
1	Support Vector Machine	99.24	99.25
2	Logistic Regression	99.25	99.21
3	KNN	99.44	99.11
4	Decision Tree	99.46	99.03
5	Naive Bayes	96.00	96.45


```
# Yapılan karşılaştırma grafiksel olarak gösterildi.

plt.subplots(figsize=(12,8))
ax=sns.barplot(x='Gözetimli öğrenme modelleri',y='Accuracy_score',data=sonuc_df)
labels = (sonuc_df["Accuracy_score"])
# add result numbers on barchart
for i, v in enumerate(labels):
    ax.text(i, v+1, str(v), horizontalalignment = 'center', size = 10)
```



SONUÇ: Altı farklı gözetimli öğrenme algoritmalarının arasında en yüksek Accuracy_score'a sahip olan modelin %99.32 ile Random Forest olduğu görülmüştür.

```
# Performansın en yüksek olduğu algoritma üzerinde yüksek öneme sahip olan özellik belirlendi.
rfc.feature_importances_
ÖzellikSeçimi=pd.DataFrame({'Feature names':X.columns,'Importances':rfc.feature_importances_})
ÖzellikSeçimi
ÖzellikSeçimi_1=ÖzellikSeçimi.sort_values(by='Importances',ascending=False)
ÖzellikSeçimi_1
```

	Feature names	Importances
2	Light	0.680992
0	Temperature	0.173653
3	CO2	0.093409
5	Nem + Nem oranı	0.021114
4	HumidityRatio	0.019346
1	Humidity	0.011487

```
# Yüksek olan özellikten düşük olana doğru grafik çizdirildi.
plt.bar(ÖzellikSeçimi_1['Feature names'],ÖzellikSeçimi_1['Importances'])
plt.show()
```

