GAZİ ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

CENG459 Special Topics in Computer Engineering

PROJE ÖDEVİ

Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models

CAN KORKUT

141180046

KASIM 2019

ANKARA

İÇİNDEKİLER

ÖZET

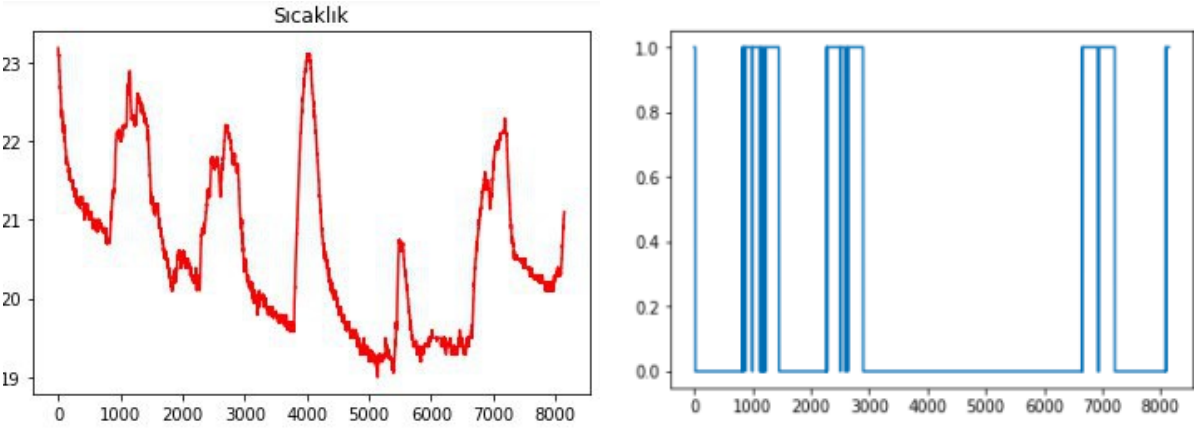
Söz konusu makalede kapalı bir ofis odasında çeşitli sensörleri kullanarak oda içerisinde herhangi bir kişinin bulunup bulunmadığı kontrol edilmekte. Bu çalışmanın yapılma amacı ise doluluk durumuna göre enerji tasarfunun mümkün olmasıdır. Eğer oda içerisinde kimse bulunmuyor ise ışıklar otomatik kapatılabilir, sıcaklık düşürülebilir ve bunun benzeri eylemler yapılarak tasarruf sağlanabilir. Bu durumun gerçekleştirilebilmesi oda içerisinde belirli bir süre ışık akısı, sıcaklık, nem ve karbondioksit seviyesi bilgileri kayıt altına alınmıştır. Bu verilerin toplanabilmesi için gerekli sensörler rasperriy pi ortamında kullanılmıştır. Bunun yanında odanın dolu olup olmadığı kamera vasıtası ile belirlenmiştir. Çalışma boyunca CART, RF, GBM ve LDA istatistik modelleri kullanılmıştır. Elde edilen sonuçlarda verilerin ikili kombinasyonlarında verimli sonuç alınmıştır (CO2 ve Işık, Nem ve Sıcaklık, Sıcaklık ve CO2 vb.).

GİRİŞ

Bu rapor CENG459 Special Topics in Computer Engineering dersinin birinci proje ödevi kapsamında verilen ‘ Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models ’ adlı makalenin araştırılması ve supervised learning algoritmalarının gerçek hayat verileri ile kullanılması çalışmasını detaylı anlatmak için hazırlanmıştır. Rapor süresince ilk olarak verilen makalenin özeti ve değerlendirilmesi yer almaktadır. Bunun yanı sıra literatürde bulunan benzer çalışmalarda incelenmiştir. Raporun bir sonraki aşamasında ise proje kapsamında verilen verilerin analiz edilmesi ve değerlendirilmesi yer almaktadır. Raporun son aşamasında ise veri seti kullanılarak oluşturulan modeller ders kapsamında öğrenilen supervised algoritmaları ile eğitilmiş ve sonuçlar incelenmiştir.

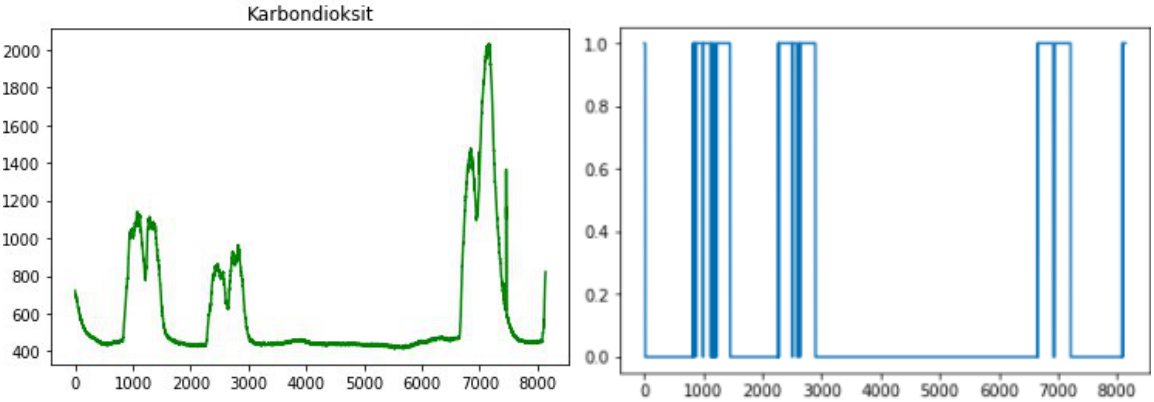
VERİLERİN ANALİZİ

Proje kapsamında kullanılmak üzere CO2 seviyesi, ışık seviyesi, sıcaklık ve nem seviyesi bilgileri ve oda doluluk durumu verilmiştir. İlk aşamada bu verilerin odanın doluluk durumu ile durumunu anlamak için grafik üzerinde incelenmiştir.



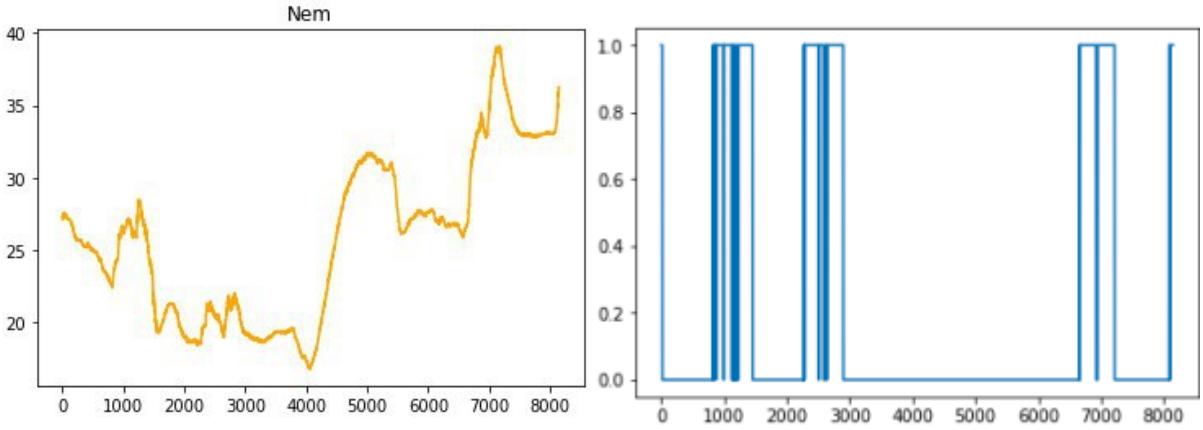
Şekil 1: SICAKLIK DOLULUK

Yukarıdaki şekilde görüldüğü üzere oda dolu iken sıcaklık artmakta ve yüksek seviyelere ulaşmaktadır. Fakat bunun yanında odanın boş olduğu süre zarfında da sıcaklık artıı olmaktadır.



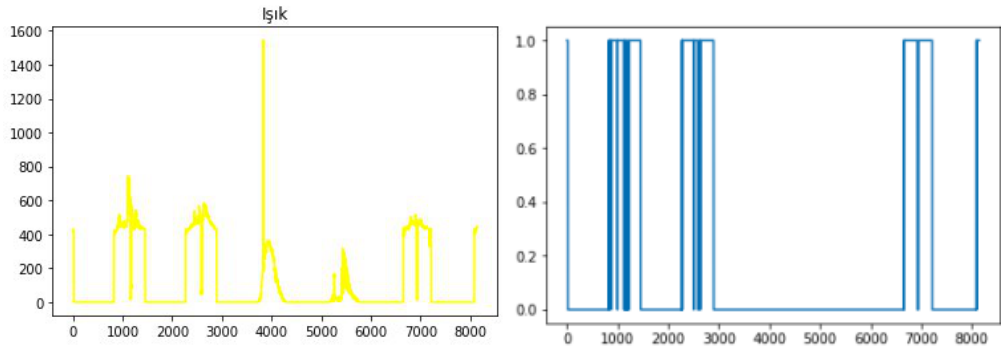
Şekil 2: CO2 DOLULUK

Sıcaklık verisinin aksine karbondioksit verisi oda boş iken çok küçük değişimlerde bulunmuştur. Bunun yanında oda dolu ilen artmaya boş iken azalmaya başlamıştır.



Şekil 3: NEM DOLULUK

Diğer iki verinin aksine nem verisi ile odanın doluluk oranı arasında lineer bir bağlantı bulunmamakta. Oda boş iken nem verisinin hızla yükseldiği görülmekte.



Şekil 4: IŞIK DOLULUK

Diğer verilerden farklı olarak ışık seviyesi verisi daha düzenli bir yapıdadır. Bunun yanı sıra odanın doluluk durumu ile belli bir seviye doğrusallık göstermektedir. Yukarıda şekilleri bulunan dört veri incelenmiştir. İncelene sonucunda ışık, CO2 ve sıcaklık verilerinin odanın doluluk oranı ile doğrusallık gösterdiği saptanmıştır. Nem verisi ise odanın doluluk oranı ile parallelik göstermekmektedir. Bunun yanında ışık ve CO2 verileri birbirleri arasında benzerlik göstermektedir.

Zaman Verisi

Yukarıda incelenen dört verinin yanı sıra zaman ve tarih veriside bulunmaktadır. Fakat bu verileri doğrudan modelde kullanmak el verişli değildir. Bundan dolayı zaman ve tarih verileri işlenmiştir. Veri setinde tarih verisi ‘ yyyy.aa.gg – saat.dak.san’ formatında verilmiştir. Veriler incelendiği vakit tarih kısmında sadece gün verisi değişiklik göstermektedir. Aşağıda verilen kod parçası ile tarih ve saat ayrılarak iki ayrı veri olarak kullanılmıştır.

```
import datetime
def get_time(DT):
    time = str(DT.hour) +  str(DT.minute)
    time = int(time)
    return time
```

```
for i in range (0,len (datatrain["date"])):
    DT = pd.to_datetime(datatrain['date'][i])
    day = DT.day
    time = get_time(DT)
    datatrain["ID"][i] = day % 7
    datatrain["date"][i] = time
```

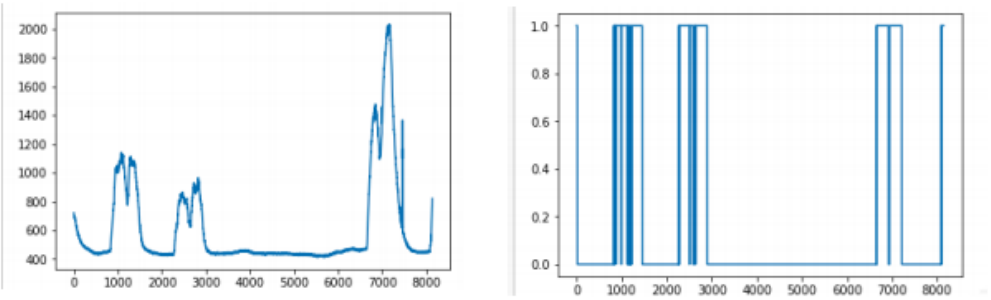
```
datatrain.rename(columns={'ID': 'Day'}, inplace=True)
datatrain.rename(columns={'date': 'Time'}, inplace=True)
```

Şekil 5: Zaman Verisi Ön İşleme

Yukarıda bulunan kod parçasında ilk olarak tarih kısmında bulunan saat ve dakika verileri ayrıştırılıyor. Daha sonra veri uzunluğunda döngü içinde sırası ile tüm veriler okunarak tarih kısmında sadece gün verisi ayrıştırılıp mod 7 değeri kayıt ediliyor. Son olarak yeni bir sûtun eklenerek saat verisi oraya kayıt ediliyor.

Yeni Verinin Oluşturulması:

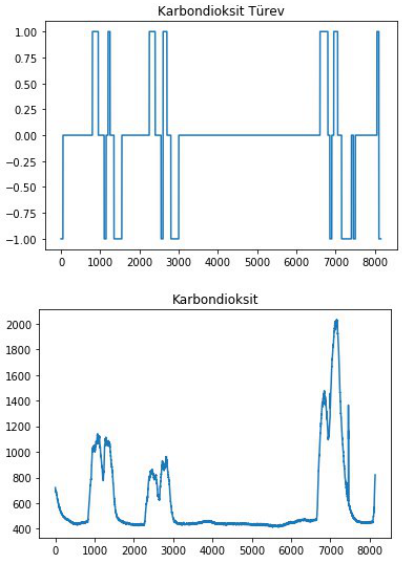
Proje kapsamında verilen veriler incelendiğinde karbodioksit verisinin odanın doluluk durumu ile parallelik göstermesinin yanı sıra odanın durum değiştirmesinden de etkilenmektedir. Oda boş durumdan dolu duruma geçtiğinde artmaya, dolu durumdan boş duruma geçtiğinde ise azalmaya başlamaktadır.



Şekil 6: Co2 Doluluk

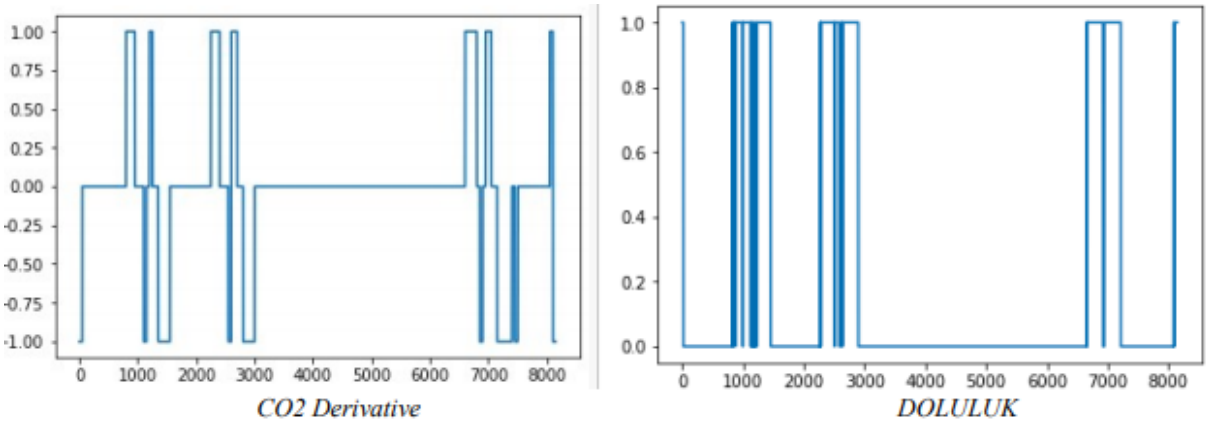
Yukaırdaki şekilde görüleceği gibi 2000 – 3000 aralığında oda boş durumdan dolu duruma geçmektedir. Bu durumda CO2 oranı artma eğilimi göstermektedir. 7000- 8000 aralığında ise oda dolu durumdan boş duruma geçmektedir, bu durumda ise CO2 azalma eğilimi göstermektedir.

Bu verinin yanı sıra 1000 – 3000 değerleri arasında oda dolu olmasına rağman CO2 seviyesi 800 – 1000 aralığında bulunurken 7000 – 8000 aralığında CO2 seviyesi 1500 – 2000 aralığında bulunmaktadır. Bu durum bize CO2 seviyesinin değerinden ziyade artma – azaşma eğiliminin önem arz ettiğini göstermektedir. Bu durumdan yola çıkarak CO2 grafiğinin kısmi türevinin alınması ile artma – azalma durumu elde edilerek yeni bir veri oluşturulabilir.



Şekil 7: CO2 Derivative CO2

Yukarıdaki grafikte CO2 seviyesi ve CO2 seviyesinin türev grafiği birlikte görülmektedir. Grafiklerden anlaşılacağı üzere CO2 seviyesi artarken grafik 1, azalırken -1 değişim göstermediği vakitlerde ise 0 değerini almaktadır. Örnek olarak 3000 – 6000 aralığında CO2 seviyesi sabit aralıktadır. Bu aralıkda CO2 türev grafiği 0 değerini göstermektedir. Bu durum odada değişiklik olmadığı anlamını taşımaktadır.



Şekil 8: CO2 Derivative DOLULUK

```
from scipy import poly1d

polinom = poly1d(datatrain['CO2'])
```

```
CO2_Deriv = poly1d.deriv(polinom)
```

```
CO2_Deriv = []
a = 1
b = 1
i = 0
while i < 8143:
    a = datatrain['CO2'][i:i+50]
    sum_a = sum(a)
    b = datatrain['CO2'][i+50:i+100]
    sum_b = sum(b)
    i = i +50
    t =  (sum_b - sum_a) / sum_a
    print t
    for x in range(0,50):
        #l.append(t)
        if t < -0.1:
            CO2_Deriv.append(-1)
        elif t > 0.1:
            CO2_Deriv.append(1)
        else:
            CO2_Deriv.append(0)
```

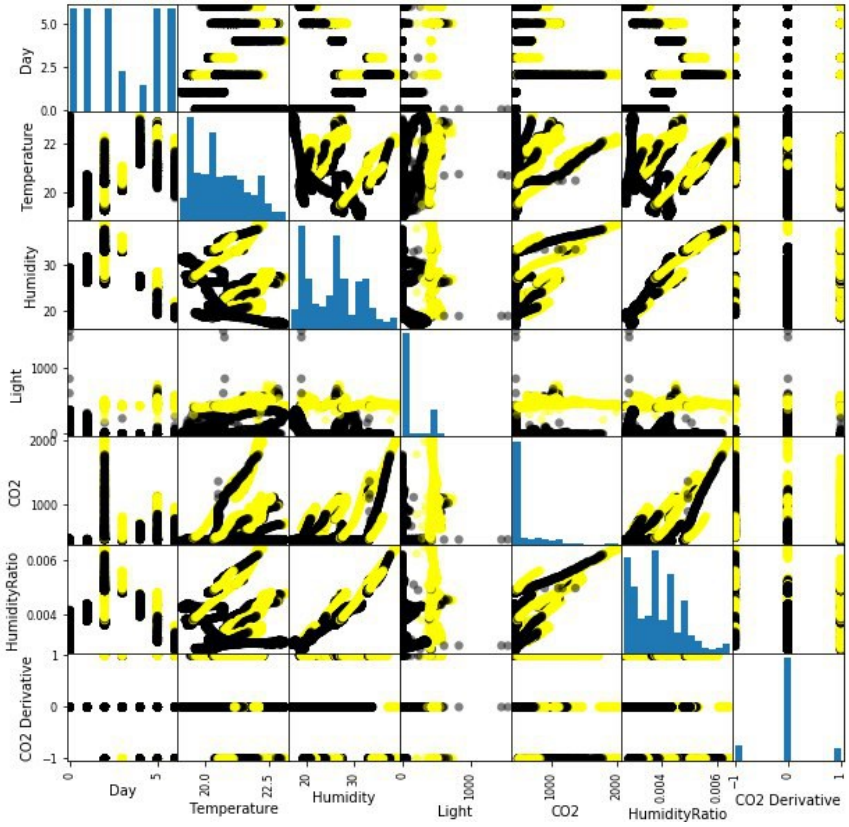
Şekil 9: Türev İşlemi

Yukarıdaki şekilde türev alma işleminin nasıl gerçekleştiği yer almaktadır. İlk olarak CO2 verisi bir polinom değişkenine atılmaktadır. Daha sonra bu veri 50 bant genişliğinde toplanarak fark alınmaktadır. Çıkan fark 0.1 (hassasiyet) ‘ den küçük ise azalmakta , -0.1’ den büyük ise artmakta her iki aralıkta ise sabit durmaktadır.

	Day	Time	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	CO2 Derivative
0	4	1751	23.180000	27.272000	426.0	721.250000	0.004793	1	-1
1	4	1751	23.150000	27.267500	429.5	714.000000	0.004783	1	-1
2	4	1753	23.150000	27.245000	426.0	713.500000	0.004779	1	-1
3	4	1754	23.150000	27.200000	426.0	708.250000	0.004772	1	-1
4	4	1755	23.100000	27.200000	426.0	704.500000	0.004757	1	-1
5	4	1755	23.100000	27.200000	419.0	701.000000	0.004757	1	-1
6	4	1757	23.100000	27.200000	419.0	701.666667	0.004757	1	-1
7	4	1757	23.100000	27.200000	419.0	699.000000	0.004757	1	-1
8	4	1758	23.100000	27.200000	419.0	689.333333	0.004757	1	-1
9	4	180	23.075000	27.175000	419.0	688.000000	0.004745	1	-1
10	4	181	23.075000	27.150000	419.0	690.250000	0.004741	1	-1
11	4	182	23.100000	27.100000	419.0	691.000000	0.004739	1	-1
12	4	183	23.100000	27.166667	419.0	683.500000	0.004751	1	-1
13	4	184	23.050000	27.150000	419.0	687.500000	0.004734	1	-1
14	4	184	23.000000	27.125000	419.0	686.000000	0.004715	1	-1
15	4	186	23.000000	27.125000	418.5	680.500000	0.004715	1	-1

Şekil 10: Veri Seti

Feature Seçimi



Şekil 11: Scatter Matrisi

Yukarıda featurların scatter matrisi yer almaktadır. Scatter matrisleri değişkenlerin birbirleri arasındaki korelasyonu göstermek amacı ile kullanılır. Featurların açıklamaları aşağıda verilmiştir. Day: Day özelliği diğer 5 özellik ile korelasyon göstermektedir. Işık değişkeni ile kolaylıkla ayrılabilir bir görünüm vermektedir.

Sıcaklık: Sıcaklık değişkeni ışık haric diğer 4 değişken ile korelasyon göstermemektedir. Yüksek başarıda bir ayrımın yapılamamaktadır fakat ışık özelliği ile korelasyonu bulunmaktadır.

Nem: Karbondioksit ve ışık değişkeni ile korelasyonu bulunmaktadır. Işık: Işık değişkeni diğer tüm özellikler ile korelasyon içindedir.

CO2 Türevi: Diğer değişkenler ile korelasyonu bulunmaktadır fakat bazı noktalarda ayrım yapmak güçtür. Bu yüzden yüksek başarı elde edilemeyebilir.

```
data_label0 = ['Day','Time', 'Temperature']
X0 = datatrain[data_label0]

data_label1 = ['Day','Time', 'Temperature','CO2']
X1 = datatrain[data_label1]

data_label2 = ['CO2','CO2 Derivative']
X2 = datatrain[data_label2]

data_label3 = ['Light','CO2']
X3 = datatrain[data_label3]

data_label4 = ['Day','Time', 'Temperature', 'Humidity']
X4 = datatrain[data_label4]

data_label5 = [ 'Temperature','CO2']
X5 = datatrain[data_label5]

data_label6 = ['Temperature', 'Humidity', 'Light','CO2',"HumidityRatio"]
X6 = datatrain[data_label6]

data_label7 = ['Humidity','CO2 Derivative']
X7 = datatrain[data_label7]
```

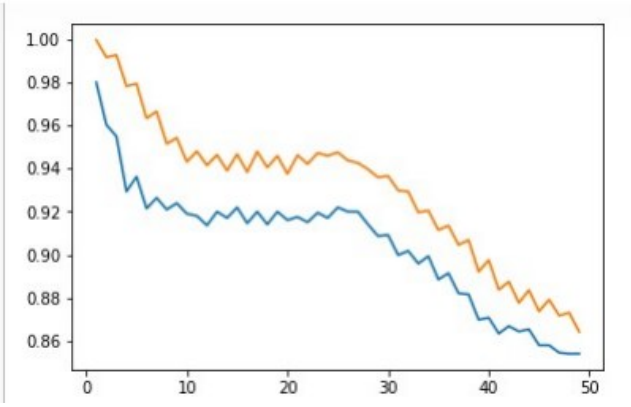
Şekil 12: Featurların Kombinasyonları

Yukarıdaki kod bloğunda modellerde denemek üzere featurların kombinasyonları oluşturulmuştur. Bu kombinasyonlar oluşturulurken bir önceki aşamada anlatılan scratte matrisindeki korelasyon durumları göz önünde tutulmuştur. Bunun yanında incelenen makalede featurların ikili kombinasyonlarının yüksek başarı sağladığı göz önünde tutulurak ikili kombinasyonlarada yer verilmiştir. Bu featurlar KneighborsClassifier algoritmasında test edilerek yüksek başarı elde edilen kombinasyon seçilmiştir.

Eğitim Aşaması

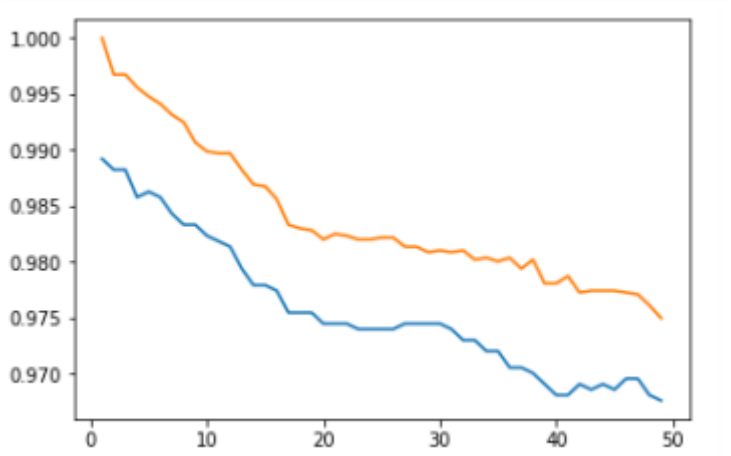
Eğitim aşamasında ilk olarak KneighborsClassifier algoritması kullanılarak feature seçimi yapılmıştır. Bunun için her kobminasyon döngüsel olarak KneighborsClassifier algoritmasında test edilmiş ve accuracy değerine göre feature seçilmiştir.

```
train_score = []
test_score = []
for i in range (1,50):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train0, y_train0)
    train_score.append(knn.score(X_train0, y_train0))
    test_score.append(knn.score(X_test0, y_test0))
plt.plot(range(1,50),test_score)
plt.plot(range(1,50),train_score)
plt.show()
```



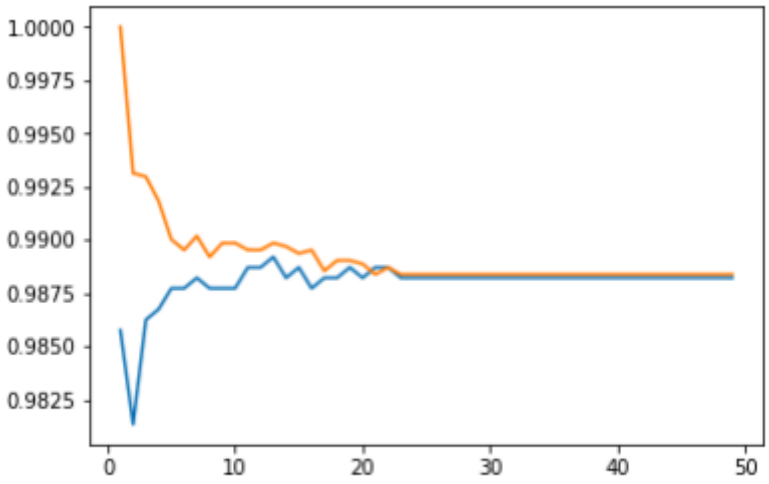
Şekil 13: Eğitim Sonucu

Yukarıdaki şekilde Gün – Saat – Sıcaklık özellikleri kullanılarak KneighborsClassifier modelinde komşuluk sayısı 1'den 50 ye kadar denenerek accuracy değerleri kayıt edilmiş ve eğitim sonucu underfitting sonucunun elde edildiği görülmüştür. Yukarıdaki şekilde Gün – Saat – Sıcaklık – CO2 özellikleri ile eğitim gerçekleştirilmiş ve eğitim sonucu underfittin gerçekleşmiştir.



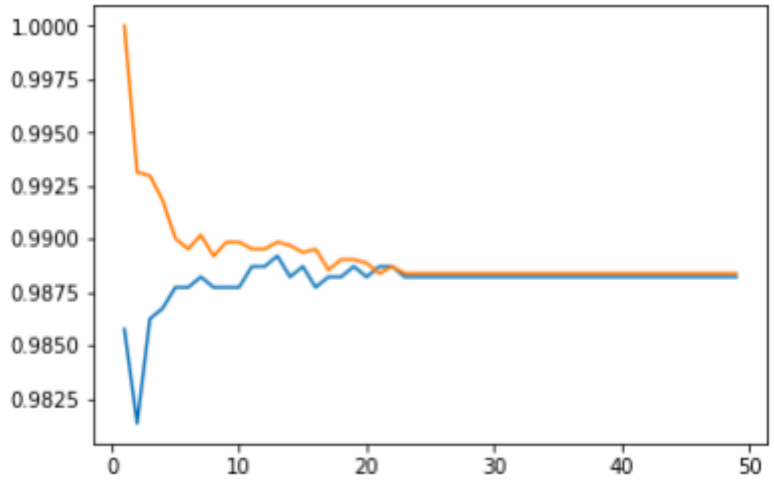
Şekil 16: Eğitim Sonucu

Yukarıdaki şekilde CO2 ve CO2 Türevi değişkenleri ile eğitim gerçekleştirilmiş komşuluk sayısı arttıkça başarı oranı artmış ve ilk iki özellik grubuna göre daha başarı sonuç elde edilmiştir. Fakat yine aynı şekilde underfittin gerçekleşmiştir.



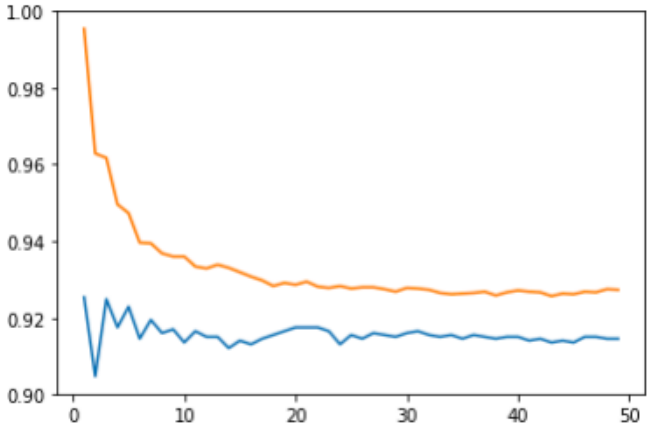
Şekil 15: Eğitim Sonucu

Yukarıdaki şekilde Işık – CO2 özellikleri ile eğitim gerçekleştirilmiş ve komşuluk sayısı 23 ve üzeri değerlerde model %99 oranında başarılı sonuç vermiştir.



Şekil 16: Eğitim Sonucu

Yukarıdaki şekilde Gün- Saat – Sıcaklık – Nem özellikleri ile eğitim yapılmış ve eğitim sonucu underfitting gerçekleşmiştir.

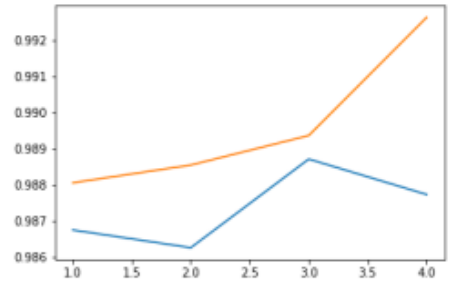


Şekil 17: Eğitim Sonucu

Yukarıdaki şekilde Sıcaklık – CO2 özellikleri ile eğitim yapılmış, komşuluk sayısının artması ile başarı oranı artmıştır fakat eğitim sonucu başarılı bir model elde edilememesine rağmen diğer özelliklere kıyasla daha başarılı bir model elde edilmiştir.

KneighborsClassifier algoritması ile yapılan testlerde " Sıcaklık – Işık " özellikleri arasında yapılan eğitimin başarı bir şekilde tamamlandığı görülmüştür. Makalede de belirtildiği gibi 2'li kombinasyonlarda daha başarılı sonuçlar elde edilmektedir.

```
from sklearn.tree import DecisionTreeClassifier
train_score = []
test_score = []
for i in range (1,5):
    clf = DecisionTreeClassifier(max_depth=i).fit(X_train6, y_train6)
    train_score.append(clf.score(X_train6, y_train6))
    test_score.append(clf.score(X_test6, y_test6))
plt.plot(range(1,5),test_score)
plt.plot(range(1,5),train_score)
plt.show()
```



Şekil 18: Eğitim Sonucu

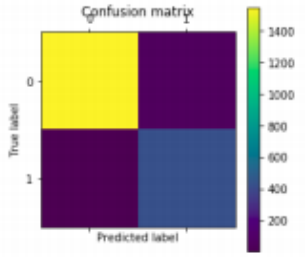
Yukarıda DecisionTreeClassifier modelinin hyper parametrelerinin ayarlanması yer almakta. Döngü içerisinde derinlik sayısı arttırılarak sonuçlar kayıt edilmiştir. Derinlik 3'ün üzerinde iken underfittin durumu oluşmaktadır. En verimli sonuç 3 ile elde edilmiştir. Aşağıda ise bazı algoritmanın sırası ile değerleri ve gerçekleştirilmesi yer almakta.

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
result = lda.fit(X_train3,y_train3).predict(X_test3)
print "accuracy score: " + str(accuracy_score(y_test3,result))
print (classification_report(y_test3,result))
cm = confusion_matrix(y_test3,result)
draw_matrix(cm)
```

```
accuracy score: 0.973968565815
             precision    recall  f1-score   support

          0       1.00      0.97      0.98      1600
          1       0.89      1.00      0.94       436

avg / total       0.98      0.97      0.97      2036
```

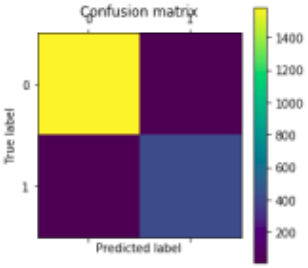



```
from sklearn.ensemble import RandomForestClassifier
random_forrest = RandomForestClassifier()
result = random_forrest.fit(X_train3,y_train3).predict(X_test3)
print "accuracy score: " + str(accuracy_score(y_test3,result))
print (classification_report(y_test3,result))
cm = confusion_matrix(y_test3,result)
draw_matrix(cm)
```

```
accuracy score: 0.988212180747
             precision    recall  f1-score   support

          0       0.99      0.99      0.99      1600
          1       0.97      0.98      0.97       436

avg / total       0.99      0.99      0.99      2036
```



Şekil 19: Confision Matrix

Sonuç

Çalışma süresince ilk olarak makale incelenerek proje hakkında bilgi edinilmiştir. İncelenilen makalede gerçekleştirilen çalışmada kapalı bir ortamda sensörler ile elde edilen ışık,CO2, nem gibi veriler eğitilerek sonuçlar elde edilmiştir. Elde edilen sonuçlarda verilerin ikili kombinasyonlarında verimli sonuç elde edilmiştir. Proje geliştirme aşamasında ise ilk olarak veriler incelenmiştir. Verilerin odanın doluluk durumu ile korelasyonu göz önünde tutulmuştur. Bir sonraki aşamada CO2 verisinin artış ya da azalışından yola çıkarak yeni bir özellik sınıfı oluşturulmuştur. Scatter matrix yöntemi ile verilerin bir birleri arasındaki korelasyonu incelenerek farklı kombinasyonlarda featurlar oluşturulmuştur. Daha sonra bu kombinasyonlar knn modelinde test edilerek en uygun featur olaral ‘ ışık – CO2 ‘ sınıfı seçilmiştir. Bir sonraki aşamada bu sınıf diğer algoritmalarda test edilerek proje tamamlanmıştır.