**İÇİNDEKİLER**

1. **Giriş**
2. **Kullanılan Veri Setinin Temini**
3. **Kullanılan Veri l Yönteminin Teorik Bilgisi**
4. **Projenin uygulanması ve performans analizi**
5. **Sonuçlar ve Değerlendirme**
6. Kaynakça

1.Giriş

Büyüme hedefi olan ve sürekli gelişmeyi hedefleyen firmaların hayatta kalarak varlığını sürdürmeleri için, yeni öngörüler, stratejiler ve en önemlisi onlara yol gösterebilecek veri analizlerine ihtiyar duymaktadırlar. Bu analizler sayesinde firmalar, yoğun rekabet ortamında hızlı ve en doğru kararı verebilir durumda olabileceklerdir. Varlıklarını devam ettirebilmek isteyen üreticiler, bir yandan artan müşteri beklentilerini karşılamak isterken bir yandan da maliyetlerini en az seviyede tutmayı amaçlamaktadırlar.

Bu çalışmada amaçlanan, önceki yılların verileri analiz edilerek, yeni bir yılın verileri hakkında ön görü sahibi olmak hedeflenmektedir.

2.Veri Setinin Temini

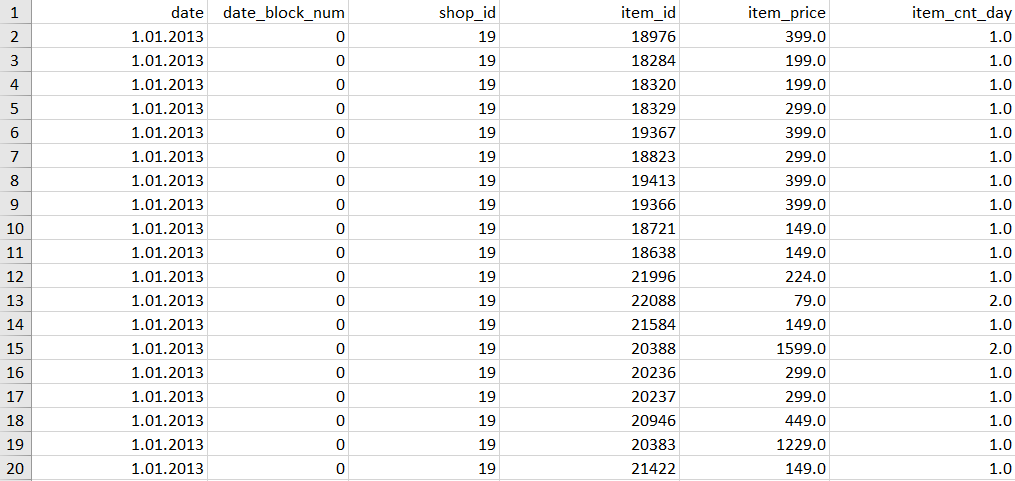
Bu çalışmada Kaggle platformunda bulunan Predict Future Sales(İleriki Zaman Satış Tahmini) veri seti kullanılmıştır. Bu veri setinde 4 adet date, shop\_id, item\_id, item\_price gibi Feature(Özellik ), 15 farklı sınıftan oluşan 1 adet item\_cnt\_day Result(Sonuç) veri sütunu kullanılmaktadır.

date: Itemin satıldığı tarihi gösterir.

shop\_id: Satılan itemin hangi mağazadan satıldığını gösterir.

item\_id: Satılan itemin Id sini gösterir.

item\_price: Satılan itemin fiyatını gösterir.

**** item\_cnt\_day: Satılan itemin kaç adet satıldığını gösterir.

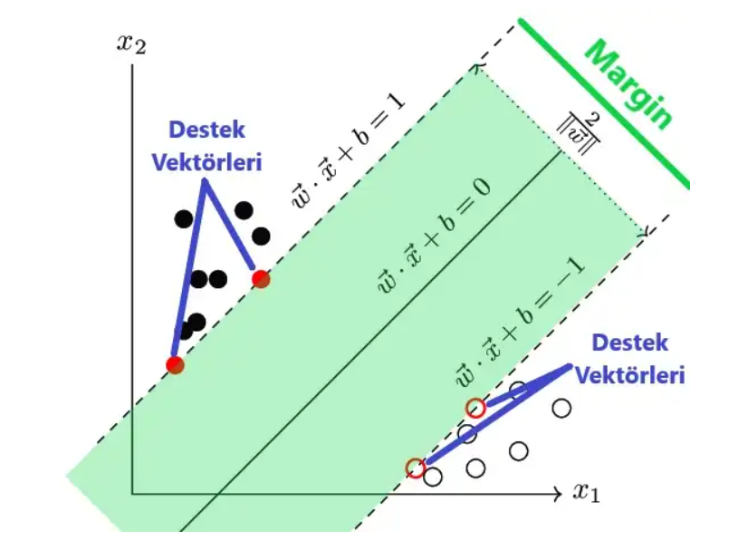
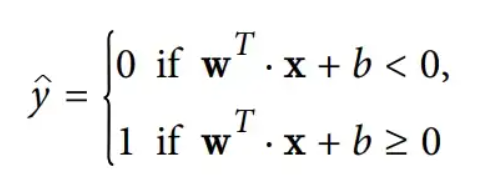
**Şekil 1** veri sütunlarına örnek

**3. KULLANILAN VERİ YÖNTEMLERİNİN TEORİK BİLGİSİ**

**3.1 Support Vector Machine (Destek Vektör Makineleri)**

Sınıflandırma problemlerinde kullanılan gözetimli öğrenme yöntemlerinden biridir.

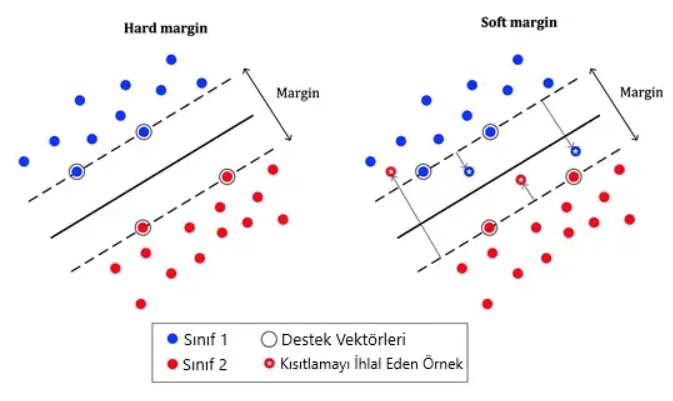
Bir düzlem üzerine yerleştirilmiş noktaları ayırmak için bir doğru çizer. Bu doğrunun, sınıfların noktaları için de maksimum uzaklıkta olmasını amaçlar.

****

**Şekil 2** SVM

SVM Formül

w; ağırlık vektörü (θ1), x; girdi vektörü, b; sapmadır (θ0). Yeni bir değer için çıkan sonuç 0'dan küçükse, beyaz noktalara daha yakın olacaktır. Tam tersi, çıkan sonuç 0'a eşit veya büyükse, bu durumda siyah noktalara daha yakın olacaktır. (**Şekil 2**)

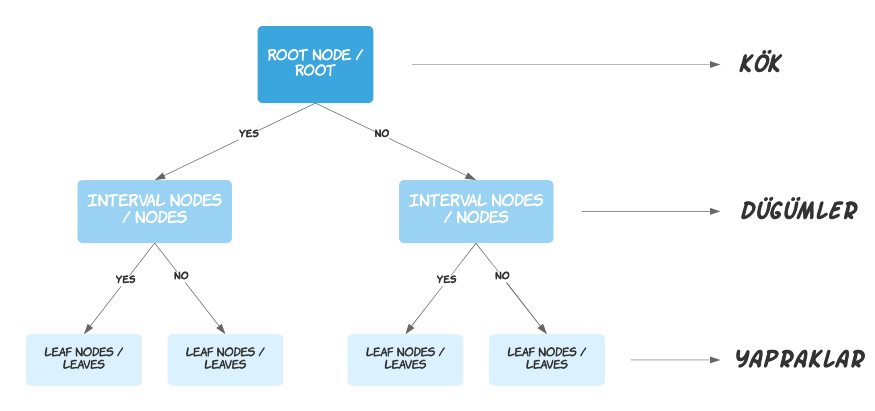
Marginimiz her zaman **Şekil 2** deki gibi olmayabilir. Bazen örneklerimiz Margin bölgesine girebilir. Buna Soft Margin **(Şekil 3.2**) denir. Hard Margin **(Şekil 3.1**), verimiz doğrusal olarak ayrılabiliyorsa çalışır ve aykırı değerlere karşı çok duyarlıdır. Bu yüzden bazı durumlarda Soft Margin’i tercih etmemiz gerekebilir.

**Şekil 3.2** Soft Margin

**Şekil 3.1** Hard Margin

**3.2 Karar Ağaçları (Decision Tree)**

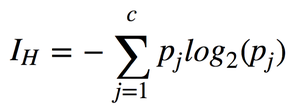
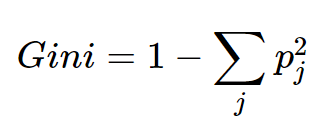
Karar ağaçları, Sınıflandırma ve Regresyon problemlerinde kullanılan, ağaç tabanlı algoritmadan biridir.



Karar ağaçlarının ilk hücrelerine “kök” denir. Her bir gözlem kökteki koşula göre “Evet” veya “Hayır” olarak sınıflandırılır.

Kök hücrelerinin altında “düğümler” bulunur her bir gözlem düğümler yardımıyla sınıflandırılır. Düğüm sayısı arttıkça modelin karmaşıklığı da artar.

Karar ağacının en altında “yapraklar” bulunur. Yapraklar, bize sonucu verir.

Kök hücre, Gini (**Şekil 4.1**)ve Entropy (**Şekil 4.2**) formülleriyle bulunur.

**Şekil 4.1** Gini Formül

**Şekil 4.2** Entropy Formül

Kök hücre bulunduktan sonra düğümler yine aynı formüller ile bulunarak yapraklara inilir.

**3.3 En Yakın Komşu (KNN)**

Tahmin edilecek değerin bağımsız değişkenlerinin oluşturduğu vektörün en yakın komşularının hangi sınıfta yoğun olduğu bilgisi üzerinden sınıfını tahmin etmeye dayanır.



**Şekil 5** Knn örnek

**Şekil 5**’e bakıldığında sarı noktanın kırmızı ve mavi noktalara olan uzaklıkları bulunur.

Bulunan değerlerin en küçük olanlarına göre yerleştirme yapılır.

**4. Projenin Uygulanması ve Performans Analizi**

Veri setimizi kullanmadan önce;

veri sayısının düzenlenmesi,

Train ve Test verisi olarak 2’ye ayrılması,

gerekli fonksiyonlarda kullanılmak üzere verilerin uygun formata koyulması gerekmektedir.

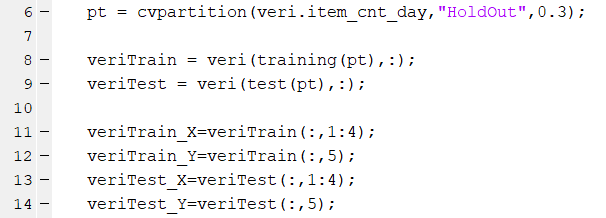
Veri setimizdeki date değerleri date türünde olduğu için , fonksiyonlarda ve algoritmalarda bu tür kullanılamadığı için numeric sayılara dönüştürülmesi gerekir.

datenum(veri.date);

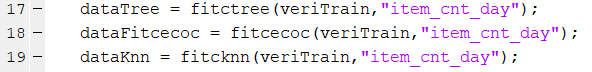
Hazırlanan verilerde değer farkları çok fazla olduğu için normalizasyon işlemine tabii tutulması gerekir.

normalize(veri);

Normalize edilen veri, Test ve Train olarak ayrılmaya hazırdır.



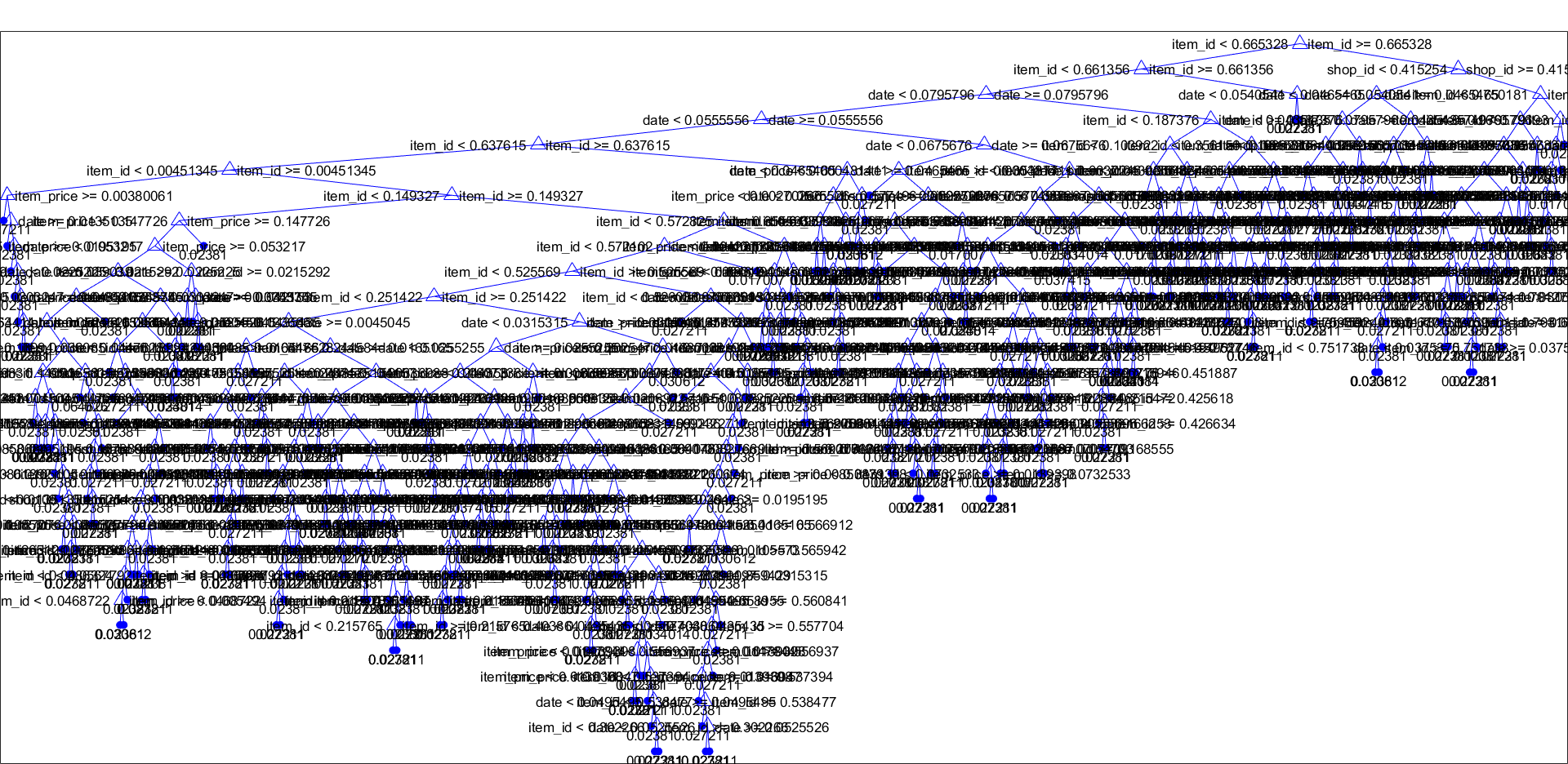
Test ve Train olarak ayrılan veri model eğitmek için hazırdır.



Oluşturulan DT modeli **Şekil 6**’da ki kod kullanılarak ağaç çizdirilir. Ve **Şekil 7**’ deki Ağaç modeli ortaya çıkar.



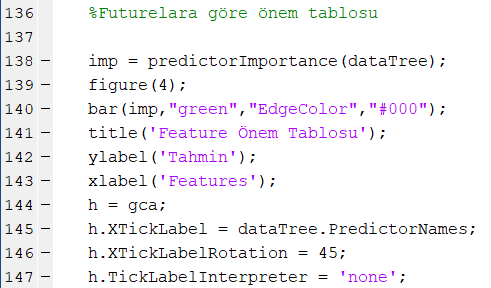
**Şekil 6** DT



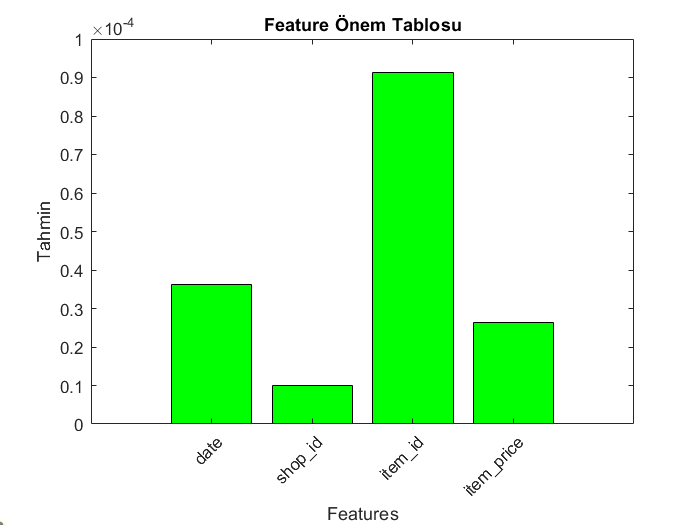
**Şekil 7** Ağaç Modeli

Ağaç Modelinin bu kadar karmaşık olmasının sebebi Result değerlerinde 15 farklı sınıftan oluşuyor olmasıdır.

DT modelini **Şekil 8.1**’de ki kodda kullanarak , veriyi işlemede kullanılan Featurların önem tablosu **Şekil 8.2** çıkarılabilir.



**Şekil 8.1** Feature önem kodu



**Şekil 8.2** Feature Önem Tablosu

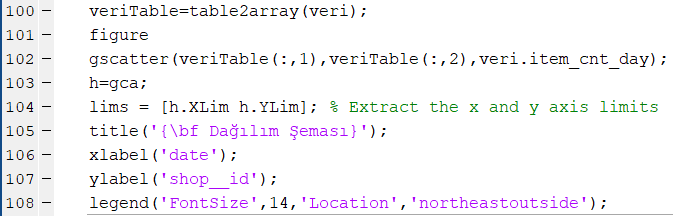
SVM modelini incelersek,

dataFitcecoc = fitcecoc(veriTrain,"item\_cnt\_day");

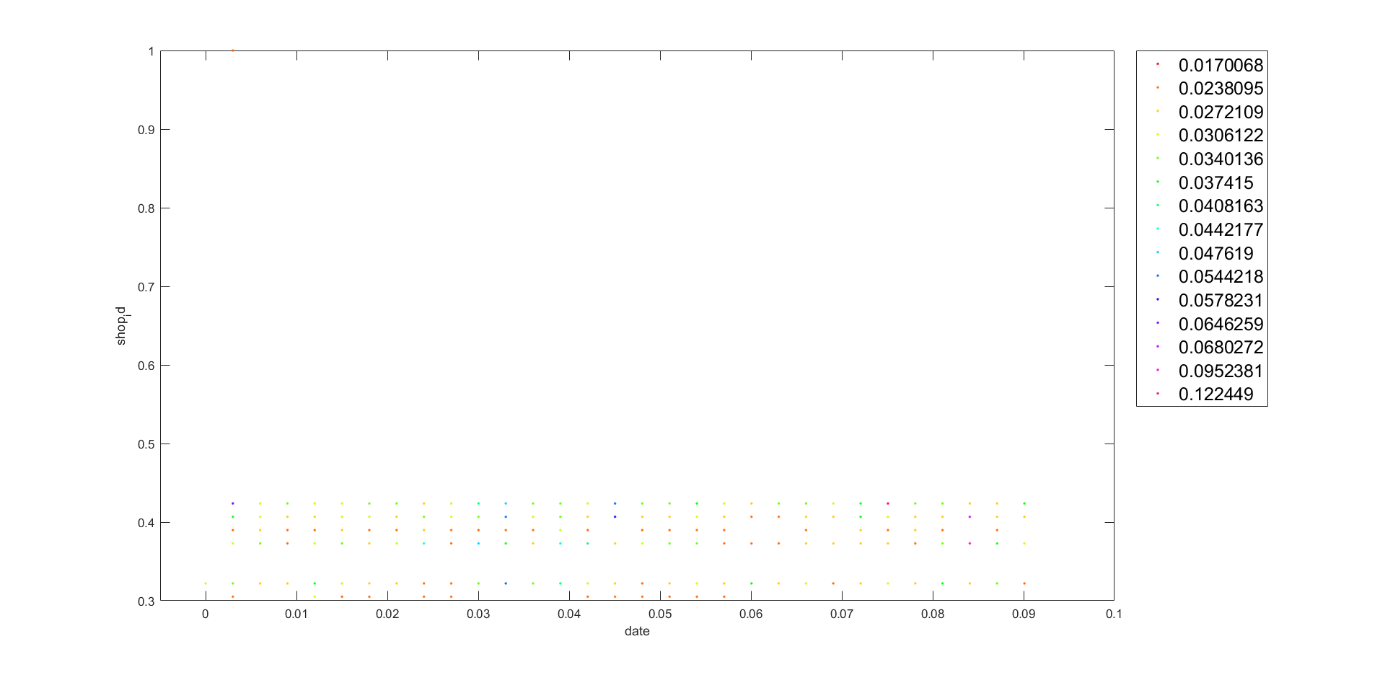
Normalde kullanılması gereken kod fitcecoc yerine fitsvm şeklinde olması gerekir.

fitsvm modeli en fazla 2 sınıflı veri setlerinde kullanılır. fitcecoc ise multi-class lı veri setlerinde kullanılması için uygundur.

SVM modelini **Şekil 9.1** de ki kodda kullanarak **Şekil 9.2** deki grafik elde edilir.

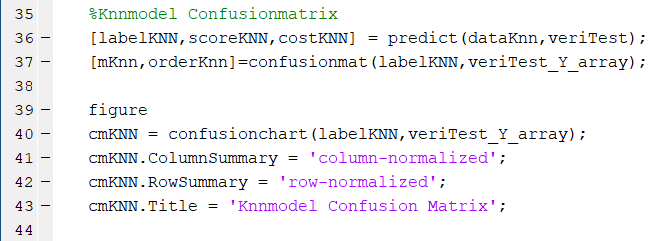


**Şekil 9.1**

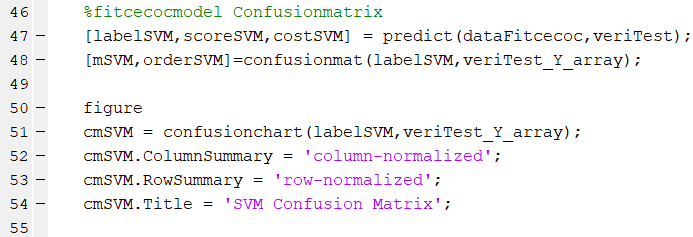


**Şekil 9.2** SVM grafik

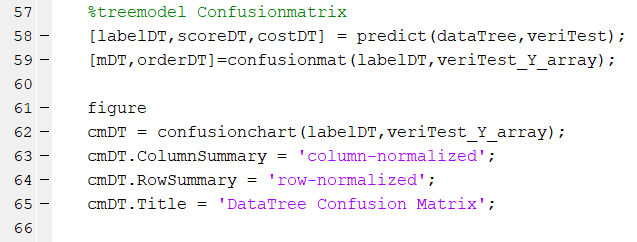
Eğitilen modeller üzerinden confusion Matrix çıkarmak için Şekil 10.1, Şekil 10.2, Şekil 10.3 daki kodlar 3 farklı model için kullanılır.



**Şekil 10.1** Knn model Confusion matrix kod

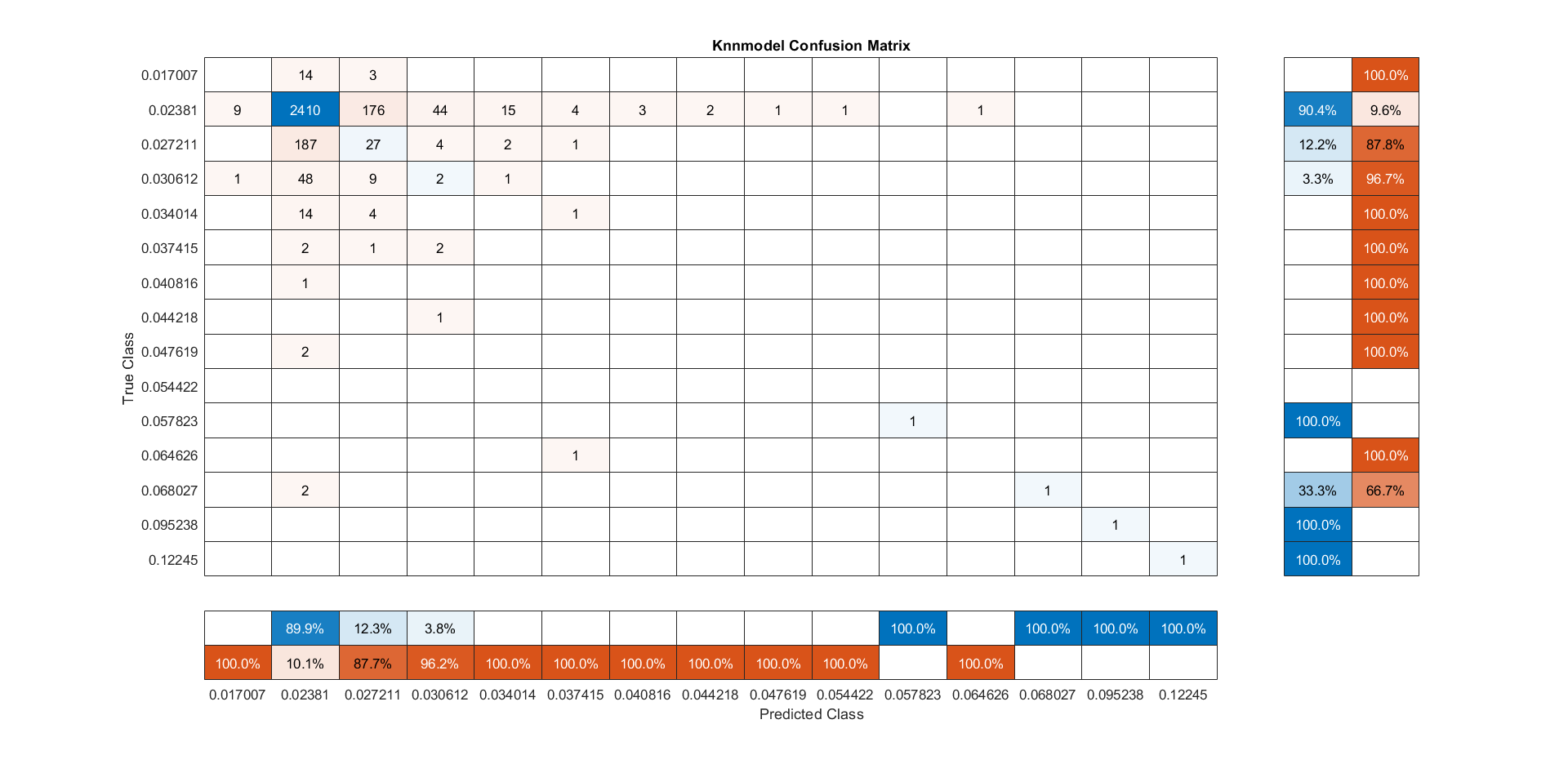


**Şekil 10.2** SVM model Confusion matrix kod

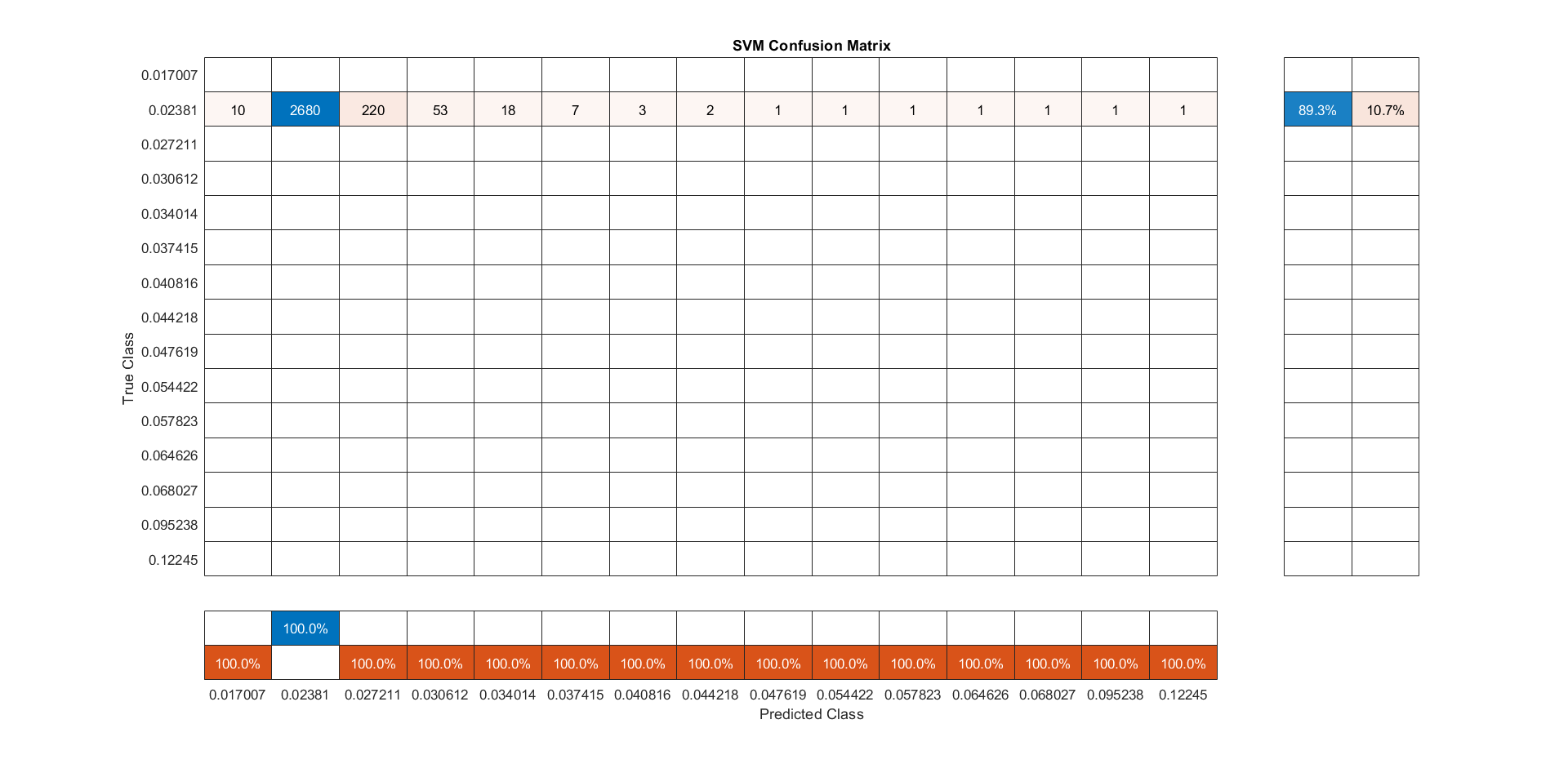


**Şekil 10.3** DT model Confusion matrix kod

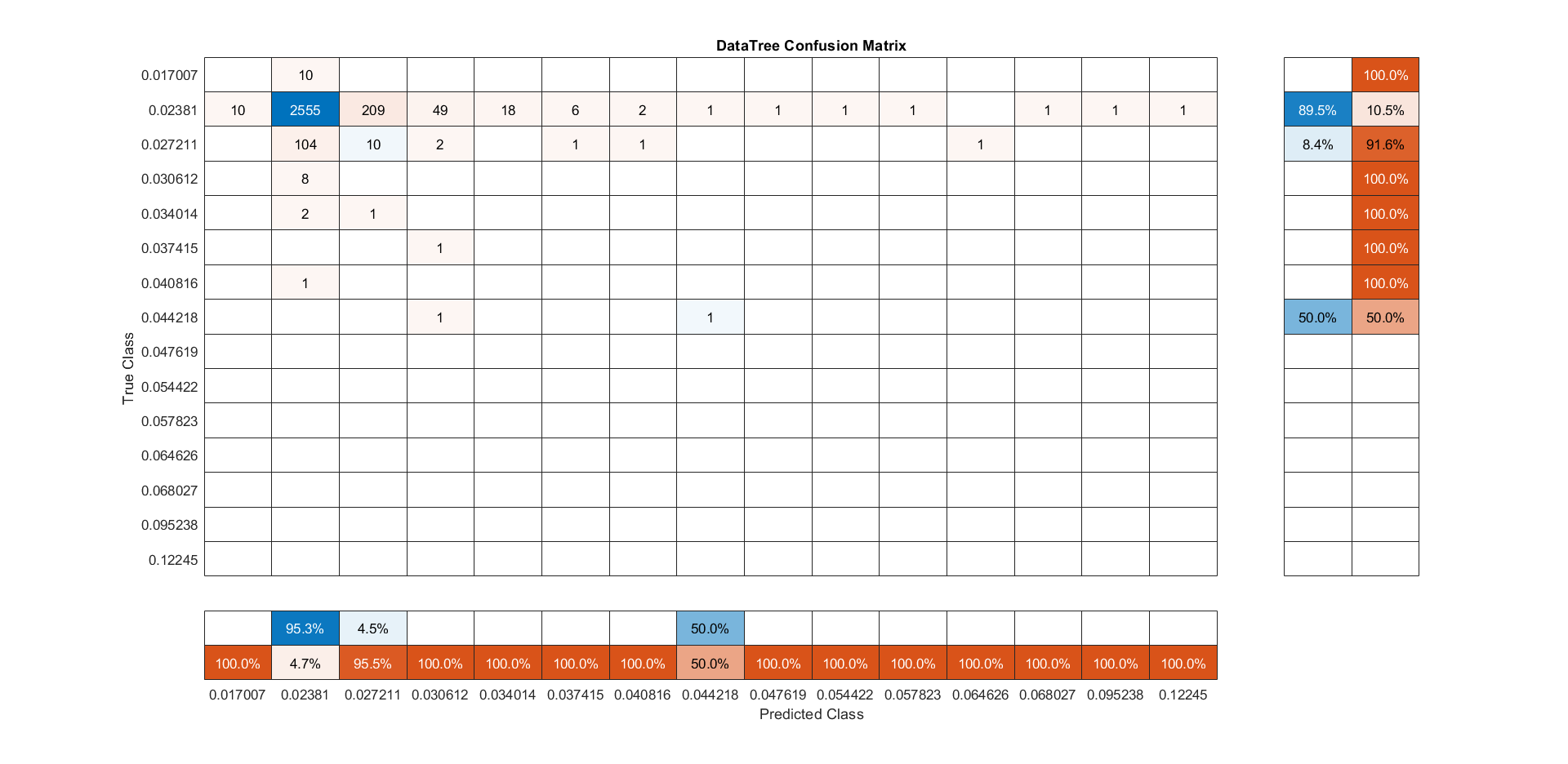
**Şekil 10.1**, **Şekil 10.2**, **Şekil 10.2** de kullanılan kodlar ile **Şekil 11.1**, **Şekil 11.2**, **Şekil 11.3**

Teki Confusion Matrixleri elde edilir.

**Şekil 11.1** Knn model Confusion matrix



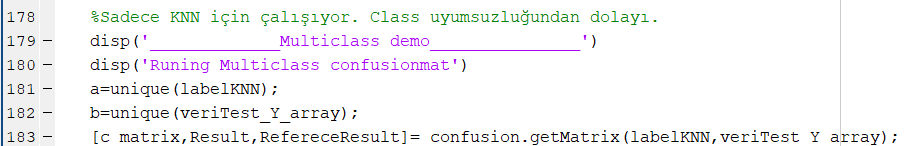
**Şekil 11.2** SVM model Confusion matrix



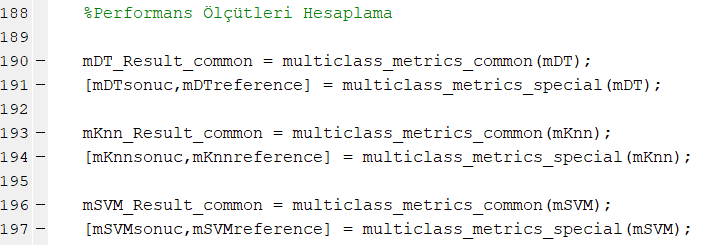
**Şekil 11.3** DT model Confusion matrix

Performans ölçütlerini hesaplamak için ,Knn modeli için 2 farklı, Dt ve SVM model için 1 tane fonksiyon kullanılmıştır.

Knn modeli için **Şekil 12** ve **Şekil 13** teki kodlar kullanılmıştır.

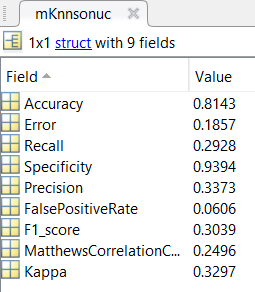
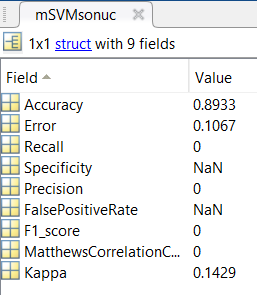
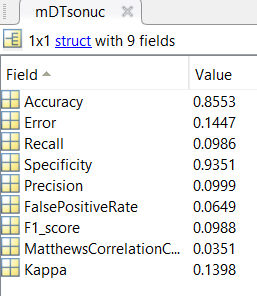


**Şekil 12** Knn özel performans ölçütü



**Şekil 13** Performans ölçütü hesaplama kod

Modellerin performans değerleri **Şekil 14.1**, **Şekil 14.2**, **Şekil 14.3** de gösterilmiştir.

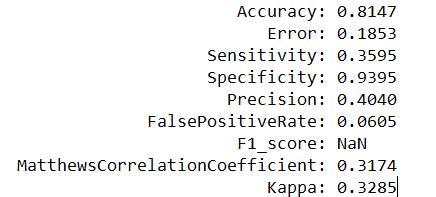


**Şekil 14.3** DT performans

**Şekil 14.1** Knn performans

**Şekil 14.2** SVM performans

Knn performans hesaplanması 2 farklı fonksiyonda yapılmıştır. Diğer fonksiyonun değerleri şu şekildedir.



**5. SONUÇLAR VE DEĞERLENDİRME**

Bu 3 modelin sonuçlarına bakıldığında Accuracy en yüksek olan SVM olarak görülür.

Ama SVM modelinde olmayan diğer modellerde olan , diğer sınıf değerleri üstünden tahmin yapması bi bakıma daha uygun gibi gözüküyor. Veri sayısı, veri oranı gibi gibi etkenler değiştirilip eklenirse daha farklı sonuçlar elde edilir.

**Şekil 14.1**, **Şekil 14.2**, **Şekil 14.3** de kullanılan fonksiyon aşağıda verilmiştir.

% Size of confusion matrix

[row,col]=size(confMatrix);

if row~=col

error('Confusion matrix dimention is wrong')

end

% Number of classes

n\_class=row;

% Find TP, FN, FP and TN

switch n\_class

case 2

TP=confMatrix(1,1);

FN=confMatrix(1,2);

FP=confMatrix(2,1);

TN=confMatrix(2,2);

otherwise

TP=zeros(1,n\_class);

FN=zeros(1,n\_class);

FP=zeros(1,n\_class);

TN=zeros(1,n\_class);

for i=1:n\_class

TP(i)=confMatrix(i,i);

FN(i)=sum(confMatrix(i,:))-confMatrix(i,i);

FP(i)=sum(confMatrix(:,i))-confMatrix(i,i);

TN(i)=sum(confMatrix(:))-TP(i)-FP(i)-FN(i);

end

end

%% Calulations

%1.P-Positive

%2.N-Negative

%3.acuuracy

%4.error

%5.Recall (Recall or True positive rate)

%6.Specificity

%7.Precision

%8.FPR-False positive rate

%9.F\_score

%10.MCC-Matthews correlation coefficient

%11.kappa-Cohen's kappa

P=TP+FN;

N=FP+TN;

switch n\_class

case 2

accuracy=(TP+TN)/(P+N);

Error=1-accuracy;

Result.Accuracy=(accuracy);

Result.Error=(Error);

otherwise

accuracy=(TP)./(P+N);

Error=(FP)./(P+N);

Result.Accuracy=sum(accuracy);

Result.Error=sum(Error);

end

ReferenceResult.AccuracyOfSingle=(TP ./ P)';

ReferenceResult.ErrorOfSingle=1-ReferenceResult.AccuracyOfSingle;

Recall=TP./P;

Specificity=TN./N;

Precision=TP./(TP+FP);

FPR=1-Specificity;

beta=1;

F1\_score=( (1+(beta^2))\*(Recall.\*Precision) ) ./ ( (beta^2)\*(Precision+Recall) );

MCC=[( TP.\*TN - FP.\*FN ) ./ ( ( (TP+FP).\*P.\*N.\*(TN+FN) ).^(0.5) );...

( FP.\*FN - TP.\*TN ) ./ ( ( (TP+FP).\*P.\*N.\*(TN+FN) ).^(0.5) )] ;

MCC=max(MCC);

%Kappa Calculation BY 2x2 Matrix Shape

pox=sum(accuracy);

Px=sum(P);TPx=sum(TP);FPx=sum(FP);TNx=sum(TN);FNx=sum(FN);Nx=sum(N);

pex=( (Px.\*(TPx+FPx))+(Nx.\*(FNx+TNx)) ) ./ ( (TPx+TNx+FPx+FNx).^2 );

kappa\_overall=([( pox-pex ) ./ ( 1-pex );( pex-pox ) ./ ( 1-pox )]);

kappa\_overall=max(kappa\_overall);

%Kappa Calculation BY n\_class x n\_class Matrix Shape

po=accuracy;

pe=( (P.\*(TP+FP))+(N.\*(FN+TN)) ) ./ ( (TP+TN+FP+FN).^2 );

kappa=([( po-pe ) ./ ( 1-pe );( pe-po ) ./ ( 1-po )]);

kappa=max(kappa);

%%

%Output Struct for individual Classes

% RefereceResult.Class=class\_ref;

ReferenceResult.AccuracyInTotal=accuracy';

ReferenceResult.ErrorInTotal=Error';

ReferenceResult.Recall=Recall';

ReferenceResult.Specificity=Specificity';

ReferenceResult.Precision=Precision';

ReferenceResult.FalsePositiveRate=FPR';

ReferenceResult.F1\_score=F1\_score';

ReferenceResult.MatthewsCorrelationCoefficient=MCC';

ReferenceResult.Kappa=kappa';

ReferenceResult.TruePositive=TP';

ReferenceResult.FalsePositive=FP';

ReferenceResult.FalseNegative=FN';

ReferenceResult.TrueNegative=TN';

% Remove NANs and INFs

stats = [Precision', Recall', F1\_score', MCC'];

stats(any(isinf(stats),2),:) = 0;

stats(any(isnan(stats),2),:) = 0;

%Output Struct for over all class lists

Result.Recall=mean(stats(:,2));

Result.Specificity=mean(Specificity);

Result.Precision=mean(stats(:,1));

Result.FalsePositiveRate=mean(FPR);

Result.F1\_score=mean(stats(:,3));

Result.MatthewsCorrelationCoefficient=mean(stats(:,4));

Result.Kappa=kappa\_overall;

**KAYNAKÇA**

(Beyaz eşya sektöründe satış tahmini)

<https://acikerisim.uludag.edu.tr/bitstream/11452/26450/3/Ezgi_Demirer_Polat_.pdf>

(Perakende sektöründe veri madenciliği ile satış tahmini)

<https://acikbilim.yok.gov.tr/bitstream/handle/20.500.12812/59571/yokAcikBilim_10321086.pdf?sequence=-1&isAllowed=y>

(SVM Examples)

<https://www.researchgate.net/figure/An-example-of-SVM-classification-An-example-of-SVM-classification_fig1_336085357>

(Multiclass metrics of a confusion matrix)

<https://www.mathworks.com/matlabcentral/fileexchange/105825-multiclass-metrics-of-a-confusion-matrix>

(Multiclass confusion Matrix)

<https://www.mathworks.com/matlabcentral/fileexchange/60900-multi-class-confusion-matrix>

(Multiclass Fitcsvm-Fitcecoc)

<https://www.mathworks.com/help/stats/classificationecoc.html>

(SVM tutorial)

<https://www.youtube.com/watch?v=q778MSq21vU&t=1108s&ab_channel=ExploringtheMeaningOfMath>

**T.C.**

**BANDIRMA ONYEDİ EYLÜL ÜNİVERSİTESİ**

**MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ**



**Gelecekteki Satış Tahmini**

**Fatih Aydoğdu**

**191502018**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**ARALIK 2022**