İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÖĞRENMESİNDE AYRIK ÖBEKLEME VE SINIFLANDIRMA ALGORİTMALARI

YÜKSEK LİSANS TEZİ

Kerem KABİL

İMatematik Mühendisliği Anabilim Dalı

Matematik Mühendisliği Programı

ARALIK 2019

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÖĞRENMESİNDE AYRIK ÖBEKLEME VE SINIFLANDIRMA ALGORİTMALARI

YÜKSEK LİSANS TEZİ

Kerem KABİL
(509161291)

İMatematik Mühendisliği Anabilim Dalı

Matematik Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Atabey KAYGUN

ARALIK 2019

İTÜ, Fen Bilimleri Enstitüsü'nün 509161291 numaralı Yüksek Lisans Öğrencisi Kerem KABİL, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "MAKİNE ÖĞRENMESİNDE AYRIK ÖBEKLEME VE SINIFLANDIRMA ALGORİTMALARI" başlıklı tezini aşağıdaki imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Doç. Dr. Atabey KAYGUN** .....
İstanbul Teknik Üniversitesi

Jüri Üyeleri : **Doç. Dr. Serkan SÜTLÜ** .....
Işık Üniversitesi

Dr. Öğr. Üyesi. Gül İNAN .....
İstanbul Teknik Üniversitesi

Teslim Tarihi : **15 Kasım 2019**
Savunma Tarihi : **11 Aralık 2019**

Aileme, Kedi Dostlarıma ve Begüm'e...

ÖNSÖZ

Tez çalışmamda değerli bilgi ve tecrübeleriyle bana rehberlik eden hocam Sayın Doç. Dr. Atabey Kaygun' a çok teşekkür ederim. Ayrıca eğitim hayatımın her anında desteklerini hissettiğim aileme ve Begüm' e sonsuz teşekkürler.

Aralık 2019 Kerem KABİL

İÇİNDEKİLER

Sayfa

KISALTMALAR

KNN : K En Yakın Komşu
SVM : Destek Vektör Makineleri
KFCV : k - Katmanlı Çapraz Geçerleme
LOOCV : Birini Dışarıda Bırakarak Çapraz Doğrulama
WCSS : Kümeler Arası Toplam Karesel Hata
ID3 : Iteratif Dichotomiser 3
KKT : Karush-Kuhn-Tucker
TP : Doğru Pozitif
FN : Yanlış Negatif
FP : Yanlış Pozitif
TN : Doğru Negatif

SEMBOLLER

$\mathbf{X_{min}}$	**:** X özniteliğinin en küçük değeri
$\mathbf{X_{max}}$	**:** X özniteliğinin en büyük değeri
$\mathbf{D}$	**:** Bir veri seti
$\mathbf{x_i}$	**:** Bir D veri setindeki x özniteliğinin i. değeri
$\mathbf{c_i}$	**:** Bir D veri setindeki i. kümenin merkez noktası
$\mathbb{R}^{\mathbf{p}}$	**:** p boyutlu reel uzay
$\mathbf{P}(\mathbf{X})$	**:** Bir X olayının olasılığı
$\mathbf{P}(\mathbf{Y}\|\mathbf{X})$	**:** Bir X olayı gerçekleştiğinde Y olayının olma olasılığı
$\mathbf{argmax}(\mathbf{x_i})$	**:** x_i noktalarının maksimumu
$\mathbf{P}(\mathbf{X})$	**:** Bir X olayının olasılığı
$\mathbf{H}(\mathbf{S})$	**:** Bir S kümesinin entropisi
$\mathbf{H}(\mathbf{X},\mathbf{C})$	**:** C sınıf özniteliğine bağlı olarak X özniteliğinin entropisi
$\mathbf{Gain}(\mathbf{X},\mathbf{C})$	**:** Bilgi kazancı fonksiyonu
$\mathbf{GainRatio}()$	**:** Bilgi kazancı oranı
$\mathbf{SplitInfo}()$	**:** Bölme bilgisi normalizasyonu
π	**:** Başarı olasılığı
$\mathbf{E}(\mathbf{Y}\|\mathbf{x})$	**:** x verildğinde Y' nin koşullu ortalaması
$\mathbf{O}$	**:** Üstünlük. Bir olayın olma olasılığının olmama olasılığa oranı
$\mathbf{OR}$	**:** İki üstünlüğün birbirine oranı
$\mathbf{ML}(\mathbf{x})$	**:** Maksimum olabilirlik fonksiyonu
$\mathbf{H}$	**:** Hiper düzlem
$\mathbf{W}$	**:** Ağırlık vektörü
$\mathbf{M}$	**:** Marjin
$\mathbf{F}_\alpha$	**:** F-ölçütü genel ifadesi
κ	**:** Kappa skoru

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

MAKİNE ÖĞRENMESİNDE AYRIK ÖBEKLEME VE SINIFLANDIRMA ALGORİTMALARI

ÖZET

Bu tez çalışması, K-Ortalamalar *K-Means*, K-En Yakın Komşu *K-Nearest Neighbor)*, Naive Bayes, Karar Ağaçları *(Decision Trees)*, Logistik Regresyon *Logistic Regression*, Destek Vektör Makineleri *Support Vector Machines* gibi bazı makine öğrenmesi algoritmalarının matematiksel altyapısını küçük örneklerle destekleyerek kapsamlı ve anlaşılabilir bir şekilde aktarmayı, bu algoritmalar kullanılarak bir makine öğrenmesi modelinin nasıl kurulması gerektiğini, kurulan bir modelin farklı özellikteki veri setleri üzerindeki performanslarının nasıl değiştiğini ve bu performansların nasıl yorumlanmasını gerektiğini göstermeyi amaçlamaktadır.

Günümüzde makine öğrenmesinin önemi gittikçe artmaktadır. Tanımlayıcı analitik *(Descriptive analytics)* ve tanısal analitik *(Diagnostic analytics)* ile hazırlanan raporlar ve analizler artık yerini tahminleyici analitik *(Predictive analytics)* ve kuralcı analitik *Prescriptive analytics* ile hazırlanan tahminlere ve analizlere bırakmaktadır. Çünkü, insanlar artık geçmişte ne olduğundan çok gelecekte ne olacak sorularıyla ilgilenmeye başlamışlardır. Çünkü, artık geleceğin verisi daha değerli hale gelmiştir. Makine öğrenmesi ile verinin daha değerli olduğu alanlarda ileriye dönük tahminler, çıkarımlar yapılabilecektir.

Makine öğrenmesi, bir sistemin geçmişte veya anlık edindiği tecrübeleri kurulan bir model yardımıyla öğrenip, gelecekte meydana gelebilecek benzer bir olayda tahmin yapabilmesini amaçlayan bir yapay zeka alanıdır.

Makine öğrenmesi kullanılarak yapılan bir tahminin başarısı kurulan modelle doğrudan ilişkilidir. Dolayısıyla matematiksel ve istatistiksel temellere dayanan makine öğrenmesi modellerini iyi inşaa etmek oldukça önemlidir. Bu durum, iyi bir makine öğrenmesi modeli kurmak için makine öğrenmesi algoritmalarının matematiksel altyapısına hakim olma gerekliliğini doğurmaktadır. Bununla beraber üzerinde çalışılacak veriyi iyi analiz etmek de oldukça önemlidir. Veri temizleme, eksik veri kontrolü, veri dönüştürme, veri ölçeklendirme gibi veri ön işleme adımlarının veri seti üzerinde doğru bir şekilde uygulanabilmesi, hangi veri seti için hangi doğrulama yönteminin kullanılması gerektiği de iyi bir makine öğrenmesi modeli için diğer gerekliliklerdendir.

Veri setleri üzerinde farklı makine öğrenmesi modelleri kullanılabilir. Fakat her model söz konusu veri seti üzerinde aynı performansı vermeyebilir. Dolayısıyla hangi modelin söz konusu veri seti için en doğru model olduğunu değerlendirebilmek de en az iyi bir model kurabilmek kadar önemlidir. Dolayısıyla bu noktada modelin değerlendirilmesinde kullanılan performans metriklerinin iyi anlaşılması büyük önem taşımaktadır.

Bu tez çalışmasında, yukarıda belirtilen hassasiyetler göz önünde bulundurularak, makine öğrenmesi algoritmalarının matematiksel altyapıları verilmiş, bir makine

öğrenmesi modeli oluşturma süreci ve oluşturulan modelin değerlendirme süreci anlatılmıştır. Teorisi anlatılan algoritmaların modelleri, *UCI Machine Learning Repository'* den alınan ve boyut, büyüklük, veri tipi olarak farklı, sınıf değişkenleri kategorik olan üç farklı veri seti *(Mushroom, Congressional Voting Records, Tic-Tac-Toe)* üzerinde kodlanmıştır. Kodlama işlemi *Python Programlama Dili* kullanılarak *Jupyter Notebook* üzerinde yapılmıştır. Bazı çıktılar *Tableau Desktop* kullanılarak görselleştirilmiştir.

DISCRETE CLUSTERING AND CLASSIFICATIONS IN MACHINE LEARNING

SUMMARY

In this thesis, we investigate commonly used classification and clustering algorithms, K-Means, K-Nearest Neighbor, Naive Bayes, Decision Tree, Logistic Regression, Support Vector Machines, in machine learning. Our goal is to express mathematical background of such classification and clustering algorithms with supporting little examples, and explain how to use a machine learning model by using these algorithms, and show how the performance of an established model on various datasets with different properties. Also, our aim is to explain how these performances should be interpreted. In addition, since machine learning is an area that requires people from different fields of study to work together, care must be taken to explain machine learning and other topics that are subject to machine learning in a way that people with different levels of expertise can understand. Another goal of this thesis is to explain machine learning and its topics in that way.

Recent history and nowadays, importance of machine learning increasing day by day. The main reason of this is the data becoming more valuable. Nowadays, people and companies are interested in these questions, "What will happen in the future?, How can make it happen?". So, nowadays reports and analyzes by using descriptive analytics and diagnostic analytics have begun to lose their importance. In parallel, predictions reports and analyzes by using predictive analytics and prescriptive analytics are becoming more valuable day by day. Therefore, thanks to machine learning, we can now work on predictive analytics and prescriptive analytics which store more valuable data in different fields. For all these reasons, machine learning has been actively used in many areas like medicine, technology, ... As time goes on, these areas of use will increase.

Machine learning is the field of artifical intelligence that aims to enable a system to learn from past or instant experiences with the help of an established model, and to predict a similar event that may occur in the future. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly. There are generally two learning types of a machine learning models such as supervised learning and unsupervised learning. In the supervised learning models, we need to seperate dataset into training data and test data basically. K-Nearest Neighbor, Naive Bayes, Decision Tree, Logistic Regression, Support Vector Machine, ...are all supervised learning algorithms. In supervised learning, we can built prediction and classification models based on both input and output data. On the other hand, in unsupervised learning there is no need to seperate dataset into training data and test data. K-Means clustering,

Hierarchical clustering... are all unsupervised learning. In unsupervised learning, we can group and interpret data based on only input data.

The success of a prediction by using machine learning algorithms is directly related to the established model. Namely, it is very important to build well machine learning models based on mathematical and statistical fundamentals. This situation necessitates a comprehensive knowhow of the mathematical background of machine learning algorithms in order to establish good machine learning model. However, it is also important to analyze well the studied dataset. Applying data preprocessing steps such as data cleaning, missing data control, data conversion, data scaling correctly is important. Also, which validation method such as k-fold cross validation, leave one out cross validation, hold-out method, re-substitution method should be used for which dataset is another requirement for a good machine learning model. Because, different validation method may cause overfitting and underfitting problems, validation method selection is directly related to machine learning prediction or classification performance. Because some validation methods is worse run time on a some dataset, validation selection is also related to model run time.

Different machine learning models may be used on datasets, but these machine learning models may not return the same performance. Therefore, evaluating which model is the most accurate model as important as establishing a machine learning model. At this point, a good understanding of performance metrics used in the evaluation of the model is of great importance. Although, all of these performance metrics are really important, some act that different. When evaluating performance of a machine learning model, some think that evaluate only accuracy is enough. This act may be fatal when evaluating performance of a machine learning model, or comparing two or more machine learning models. Because there is not only balanced dataset in the world, we also need to calculate performance metrics other than accuracy. Precision, recall, f-measure are another commonly used performance metrics in machine learning world. Each of these has different importance for the model. Except those, we investigate another performance metric in this thesis. This is called Cohen Kappa Score or Cohen Kappa Statistic. The importance of the Cohen Kappa Score is to measure whether accuracy depends on chance or not. Because this metric validate accuracy in a way, this makes it a really important performance metric. So, all performance metrics in machine learning world has different importance and meanings. Therefore, we calculate all the performance metrics when we need to evaluate performance of a machine learning model. We also calculate all the performance metrics to compare different machine learning models, and to determine which machine learning model is the best.

In this thesis, considering the sensitivities mentioned above, mathematical backgrounds of machine learning algorithms are given, process of a machine learning model and evaluation process of the model is explained. Also, in order to better explain the logic of the investigated algorithms we present fully worked out short examples for each of the algorithms we cover in this thesis. In the last chapter, we apply these algorithms on different datasets taken from *UCI Machine Learning Repository*, and analyze their performances by evaluating performance metrics values for each algorithm on this dataset. These datasets are different based on instance number, data types and dimension. In this thesis, all machine learning model is built by using k-fold cross validation. Because seeing performance scores of a machine learning model

on each cross validation steps give us some information about the performance of each fold, we are not calculate only average performance scores of machine learning models on each datasets, but also calculate performance score on each cross validation step. Also, in the last chapter Cohen Kappa Score and accuracy metric is compaired. By this comparison, it is tried to be stated that Cohen Kappa Score is a validation of accuracy metric. Thus, we can better compare different machine learning models and analyze performances in-depth.

All of these machine learning algorithms on the datasets taken from *UCI Machine Learning Repository* are coded by using *Python Programming Language* on *Jupyter Notebook*, and some of the graphs are visualised by using *Tableau*.

1. GİRİŞ

Makine öğrenmesi, verilen bir sistemin veriler üzerinden elde ettiği tecrübeler yardımıyla bir model oluşturmasını ve bu model yardımıyla gelecekteki benzer bir olay karşısında bir karar vermesini sağlayan bir yapay zeka alanıdır. Bu alanda yapılan çalışmalar geniş ve sayıca çok olduğundan bir çok makine öğrenmesi algoritması geliştirilmiş ve geliştirilmeye devam edilmektedir.

Makine öğrenmesi algoritmalarıyla bir model kurulacaksa söz konusu modelin nasıl kurulduğunu, algoritmaların nasıl çalıştığını ve model kurulumu esnasında gerekli olan parametrelerin nasıl seçileceğini bilmek son derece önemlidir. Ayrıca bu bilgiler ışığında kurulacak bir modelllerin performanslarını iyi analiz edebilmek ve hangi veri setinde hangi algoritmayı ve modeli uygulamak gerektiğini bilmek de oldukça önemlidir.

Geliştirilmiş birçok makine öğrenmesi algoritması olduğundan kullanılacak model eldeki veriye göre farklılık gösterebileceği gibi farklı veri setleri ve farklı veri tiplerinde farklı başarı oranları da elde edilebilir. Bu tez çalışması kapsamında sınıf değişkenleri kategorik veri tipinde olan veri setleri üzerinde belli kümeleme ve sınıflandırma algoritmalarını inceleyecek ve performanslarını analiz edeceğiz.

1.1 Tezin Amacı

Bu tez kapsamında Makine Öğrenmesi kapsamında yer alan K-Means, K-En Yakın Komşu, Naive Bayes, Karar Ağaçları, Logistik Regresyon ve Destek Vektör Makineleri algoritmaları hakkında bilgiler verilmiş, performans ölçütleri incelenmiş ve bazı veri setleri üzerinde uygulamaları yapılmıştır. Veri setleri *UCI Machine Learning Repository'* den alınmış olup sınıf değişkenleri kategorik veri tipinde olan veri setleri kullanılmıştır.

Makine öğrenmesi günümüzde popüler bir araştırma alanıdır. Öyle ki matematik, mühendislik gibi alanların dışında eğitim gören insanların da üzerinde uğraş verdiği

makine öğrenmesi, farklı disiplinleri bir araya getiren bir alandır. Bundan dolayıdır ki aynı makine öğrenmesi modeli üzerinde çalışıp farklı katkılar sunan bir matematikçi, bir mühendis ve bir sosyolog olabilir. Dolayısıyla makine öğrenmesi konusu mümkün olduğunca her disiplinden insanın anlayabileceği bir şekilde aktarılırsa başarı daha çok artacaktır. Buna paralel olarak bu tezin amacı; makine öğrenmesi konusuna dahil olan yukarıda listesini verdiğimiz algoritmaların matematiksel altyapısını küçük örneklerle beraber adım adım anlatarak, bu algoritmaların veri seti üzerinde nasıl uygulanacağını göstermek, gerekli ise veri ön işleme aşamalarında neler yapıldığını anlatmak ve performans ölçütleri yardımıyla performanslarını analiz etmektir.

1.2 Literatür Araştırması

Literatürde olan çalışmalar incelendiğinde, kategorik sınıf değişkenlerine sahip veri setleri üzerinde uygulanan algoritmalarının performanslarının karşılaştırıldığı birçok kaynak vardır. Bunlardan Elif Kartal, [9]' da sınıflandırmaya dayalı makine öğrenmesi algoritmalarından bazılarını anlatarak bu algoritmaların performanslarını kıyaslamıştır. Buna bir diğer örnek olarak Swapna, T. ve Sravani, Y. tarafından yazılan bir çalışma olan [3] ve Payal, P., Manju, P. ve Renu, Miglani tarafından kaleme alınan [4] örnek olarak verilebilir. Makine öğrenmesi teorisi ve algoritmalarının anlatıldığı bir kaynak olan [3], Metin Bilgin tarafından yazılmıştır. Makine öğrenmesi ve veri madenciliğini bir arada anlatan bir kaynak olan [2], Jiawei Han, Micheline Kamber ve Jian Pei tarafından yazılmıştır. Çok değişkenli gözlemlere dayalı sınıflandırma algoritmalarından bahsedilen ve k ortalama kümelemenin anlatıldığı bir kaynak olan [4], MacQuenn J. tarafından yazılmıştır. [5]' de gözetimli öğrenme yöntemlerinden olan k en yakın komşu sınıflandırma algoritması Fix ve Hodges tarafından tanıtılmıştır. Bugün en çok kullanılan sınıflandırma algoritmalarından olan karar ağaçları ise J. R. Quinlan tarafından [11]' de tanıtılmıştır. Temeli 18. yüzyılda Thomas Bayes tarafından tanıtılan Bayes Teoremi' ne dayanan naive bayes sınıflandırma algoritması [6]' da detaylı bir şekilde incelenmiştir. Gözetimli öğrenme yöntemleri kapsamında incelenen bir diğer algoritma olan destek vektör makineleri Cortes C. ve Vladimir, V. tarafından [7]' de tanıtılmıştır. Kategorik sınıf değişkenleri ve öznitelikler arasındaki ilişkinin incelendiği logistik regresyon için David W. Hosmer tarafından kaleme alınan [14], Logistik regresyon teorisi ve uygulamaları üzerine yoğunlaşmıştır. Hastie, T.,

Tibshirani, R. ve Friedman, J. tarafından yazılan [6]' da gözetimli ve gözetimsiz makine öğrnmesi yöntemleri incelenmiş, bu yöntemlere dahil olan algoritmalar ele alınmıştır. Bishop, C.M. [8]' da birçok makine öğrenmesi algoritmasını bir arada sunmuş ve bunlarla ilgili örnekler vermiştir. Alpaydın, E. tarafından yazılan ve kapsamı oldukça geniş olan [9]' da makine öğrenmesine dair tüm konular ele alınmıştır. Kategorik veri sistemleri için güzel bir kaynak olan Alan Agresti tarafından hazırlanan [10]' de her ne kadar kategorik veriler üzerine yazılmış olsa da yine makine öğrenmesi algoritmaları ve bu algoritmaların çalışma mantığı detaylı bir şekilde incelenmiştir.

1.3 Tezin Bölümleri

Birinci bölümde tezin amacından bahsedilmiş olup makine öğrenmesi hakkında giriş niteliğinde genel bilgiler verilmiştir.

İkinci bölümde öncelikle veri, değişken türleri tanıtılmış ardından bir model geliştirme sürecinin en önemli adımı olan veri ön işleme konusundan bahsedilmiştir. Veri ön işleme sürecinde kullanılan yöntemler örneklerle açıklanmıştır.

Üçüncü bölümde model değerlendirme ölçütleri olan doğrulama yöntemleri anlatılmıştır. Öncesinde aşırı öğrenme, az öğrenme durumlarından bahsedilerek model karmaşıklığı konusuna değinilmiştir.

Dördüncü bölümde ise anlatılan farklı modellerin performaslarını analiz edebilmek adına kullaılan metriklerden doğruluk, kesinlik, duyarlılık, belirleyicilik, F-Ölçütü ve Cohen Kappa Skoru metriklerinden söz edilmiştir. Ayrıca bazı durumlarda modelin başarısını değerlendirirken neden sadece doğruluk değerine bakmanın yetersiz olabileceğinden bahsedilmiştir.

Beşinci bölümde öncelikle Makine Öğrenmesi Nedir? Gözetimli ve Gözetimsiz Öğrenme Nedir? sorularına cevap verilmiştir. Ardından sırasıyla K-Means, K-En Yakın Komşu, Naive Bayes, Karar Ağaçları ve Destek Vektör Makineleri algoritmaları anlatılmıştır.

Altıncı bölümde ise teorisi verilen algoritmalar srasıyla *UCI Machine Learning* bünyesinde yer alan *Mushroom, Congressional Voting Records ve Tic-Tac-Toe* veri setleri üzerinde uygulanmış ve performansları incelenmiştir.

Yedinci bölümde altıncı bölümde yapılan deneylerin sonuçları analiz edilmiştir.

2. VERİ TÜRLERİ VE VERİ ÖN İŞLEME

2.1 Veri, Değişken Türleri

Bir araştırma ya da bir olay sonucunda elde edilmiş bilgilerin tümüne *veri* denir. Değişken ise verinin sahip olduğu öznitelikler olarak tanımlanabilir. Örneğin; bir araba veri seti düşünelim. Bu araba veri setinde yer alan marka, model, renk gibi öznitelikler birer değişkendir. Veri ise genel olarak bu veri setindeki her bilgiye veriler bir isimdir.

Bir veri seti üzerinde analiz yapmadan ya da bir veri seti üzerinde bir model inşaa etmeden önce veri setini oluşturan verileri iyi tanımak gerekir. Yani bir veri setinde yer alan özniteliklerin *(attributes)* hangi değişken tipine sahip olduklarını analiz etmek oldukça önemlidir. Çünkü değişken tipine göre kullanılacak modeller farklılık gösterebilir.

Bir veri setini oluşturan özniteliklerin her biri aynı değişken tipine sahip olabileceği gibi farklı veri tiplerine de sahip olabilir [9].

2.1.1 Kategorik *(Qualitative)* değişken türleri

Nitel veri türleri olarak da adlandırılabilirler. Üzerlerinde matematiksel işlemlerin yapılamadığı verilerdir. Bir kişinin medeni durumu (bekar, evli,...), bir arabanın rengi (siyah, beyaz, ...), eğitim düzeyi (lisans, yüksek lisans, doktora) gibi öznitelikler birer kategorik veridir. Kategorik veriler iki ayrı alt başlıkta incelenir [11].

Nominal değişken

Kategorik veri tipinde olan bu veriler arasında mantıksal bir sıralama yoktur. Yani kendi içlerinde sıralanamayan verilerdir. Örneğin; siyah, beyaz, kırmızı gibi değerlere sahip bir renk özniteliği bir nominal veridir. Çünkü, siyah, beyaz, kırmızı arasında herhangi bir mantıksal sıralama yoktur. Benzer bir örneği bir insanın medeni durumunu gösteren medeni hal *(evli, bekar,...)* özniteliği için de verebiliriz.

Ordinal (Sıralı) değişken

Kategorik veri tipinde olan ordinal verilerde nominal verilerden farklı olarak değerler arasında mantıksal bir sıralama mevcuttur. Örneğin; eğitim üzeyi (lisans, yüksek lisans, doktora), madalya türü (bronz, gümüş, altın) gibi öznitelikler ordinal veri tiplerine örnek olarak gösterilebilir.

2.1.2 Nümerik *(Quantative)* değişken türleri

Nicel veri türleri olarak da adlandırılabilirler. Sayısal olarak ifade edilmelerinin yanı sıra söz konusu sayılar arasındaki farkın matematiksel olarak bir anlamı vardır. Miktar hakkında bilgi verirler. Nümerik veri türleri kendi içinde iki ayrı alt başlık altında incelenirler.

Sürekli nümerik değişken

Ölçüm, tartım ve analiz sonuncunda elde edilen ve iki değeri arası sonsuz parçaya bölünebilen değişkenlerdir. Örnek olarak, sıcaklık, ağırlık, boy, kilo, yağış miktarı,... verilebilir.

Süreksiz nümerik değişken

Süreksiz nümerik değişkenler, sürekli nümerik değişkenlerin aksine tanım aralıklarında her değeri alamayan nümerik değişkenlerdir. Bu tür değişkenler tam sayı olarak ifade edilirler. Örnek olarak, ailedeki birey sayısı, uzayda yer alan gezegen sayısı, bir şirketteki çalışan sayısı,... verilebilir.

2.2 Veri Ön İşleme *(Data Preprocessing)*

Bir makine öğrenmesi modelinin iyi bir sonuç verebilmesi çalışılan verinin kalitesiyle direkt ilişkilidir. Veri ön işleme olarak adlandırılan bu bölüm, bir makine öğrenmesi modelinin veriden optimum şekilde yararlanabilmesi için kullanılan yöntemler bütünüdür. Genel olarak bir veri ön işleme süreci

- Eksik Veri ve Veri Temizleme (Data Cleaning)
- Veri Dönüştürme (Data Transformation)
- Veri Ölçeklendirme, Normalizasyon (Data Scaling)

adımlarından oluşur.

2.2.1 Eksik veri ve veri temizleme

Üzerinde çalışma yapılacak olan veri setinin eksik veya yanlış öznitelik değeri olarak adlandırılan gürültülü *(noisy)* veri barındırması durumunda modelden elde edilen sonuçlarda tutarsızlıklar görülebilir. Bu gibi durumlarda modelin daha iyi bir sonuç verebilmesi adına eksik veriler ve gürültülü veriler üzerinde bir takım işlemler yapmak gerekir. Bu işlemler aşağıda belirtilmiştir:

- Eksik verinin yer aldığı satır veri setinden silinebilir.
- Eğer eksik veri ve gürültülü veri nümerikse o özniteliğe ait değerlerin ortalaması alınıp eksik veri ve gürültülü veri yerine yazılabilir. Eğer öznitelik kategorik bir veri ise o öznitelikte frekansı çok olan değer eksik veri ve gürültülü veri yerine yazılabilir.
- Eksik veri için karar ağacı *(decision tree)* ya da regresyon *regression* modeli kurularak eksik veri tahmin edilebilir.

2.2.2 Veri dönüştürme

Makine öğrenmesi modellerinin doğru çalışabilmesi için kategorik veri tipine sahip öznitelik değerlerinin nümerik veri temsillerine dönüştürülmeleri gerekir. Bunun için iki farklı yöntem kullanılır:

2.2.2.1 Etiket kodlama (Label encoding)

Bir kategorik öznitelikteki her bir değere sayısal bir temsil değeri atanır. Genelde iki farklı kategorik değere *(True/False,...)* sahip öznitelik için kullanılır.

İsim	Yaş	Cinsiyet
A	25	Erkek
B	27	Erkek
C	32	Kadın

→

İsim	Yaş	Cinsiyet
A	25	1
B	27	1
C	32	0

Şekil 2.1 : Etiket kodlama işlemi.

Şekil (2.1)' de verilen örnek veri setinde Cinsiyet özniteliğinin verileri sayısal veri tipine dönüştürülmüştür. Cinsiyet özniteliği iki farklı değerli bir kategorik veri olduğu için bu durum bize sorun yaşatmayacaktır. Eğer biz bu işlemi İsim özniteliğine uygulasaydık $A \implies 0$, $B \implies 1$, $C \implies 2$ değerlerini alacaktı. Fakat bu durum modeli eğitme aşamasında sıkıntı yaşamamıza neden olabilirdi. Çünkü, bu durumda sayısal olarak yüksek değere sahip bir kategori, sayısal olarak daha düşük bir değere sahip kategoriden daha değerli olarak değerlendirilebilir ki bu durum model sonucunu olumsuz yönde etkileyecektir. Bu durumdan kaçınmak adına ikiden fazla farklı kategorik değere sahip özniteliklerin dönüşümü aşağıdaki yöntemle yapılmaktadır.

2.2.2.2 Gölge değişken *(Dummy variable / One-Hot encoding)*

One-Hot Encoding olarak adlandırılan bu yöntem bir öznitelikte yer alan her bir değeri ayrı bir özniteliğe dönüştürür. Böylece her örnek hangi özniteliğe sahipse onun değeri "1" olarak atanır.

İsim	Yaş	Cinsiyet
A	25	Erkek
B	27	Erkek
C	32	Kadın

A	B	C	Yaş	Cinsiyet
1	0	0	25	1
0	1	0	27	1
0	0	1	32	0

Şekil 2.2 : Gölge değişken oluşturma işlemi.

2.2.3 Veri ölçeklendirme

Bazı veri setlerinde bazı öznitelik değerleri diğer öznitelik değerlerine göre daha geniş bir aralıkta dağılıyor olabilir. Bu nedenle çok daha büyük değerlere sahip bir öznitelik model sonucunu domine edebilir. Örneğin; bir sporcunun yaş özniteliği 18 ila 40 arasında değer alabiliyorken maaaş özniteliği 100000 ila 5000000 arasında değer alabilir. Bu iki özniteliğin değişim aralığı birbirinden oldukça farklıdır. Dolayısıyla çok daha geniş bir dağılıma sahip olan maaş özniteliğinin model sonucuna etkisi yaş özniteliğine göre çok daha fazla olacaktır. Bu istenen bir durum değildir. Özellikle K-En Yakın Komşu *K-Nearest Neighbor (KNN)* gibi uzaklık hesabına dayalı modellerde diğerlerinden çok daha büyük bir aralıkta dağılım gösteren öznitelikler model sonucunu domine eder ve bizi yanlış bir sonuca götürür. Bu durumun önüne geçmek için yani herhangi bir özniteliğin model sonucunu domine etmemesi için

öznitelikler belli bir aralığa indirgenir. Bunun için *Min-Max Normalizasyon* yöntemi kullanılabilir.

2.2.3.1 Min-Max normalizasyon

Bu yöntemde, bir özniteliğe ait değerlerden en büyük ve en küçük olanı sırasıyla 1 ve 0 olacak şekilde kalan diğer tüm değerler (0,1) aralığında normalleştirilir.

X_{min} özniteliğin en küçük değeri, X_{max} ise en büyük değeri olmak üzere bu öznitelikteki bir X değerinin $[0,1]$ aralığına indirgenmiş hali

$$X^* = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.1}$$

şeklinde hesaplanır.

3. MODEL PERFORMANSI DEĞERLENDİRME ÖLÇÜTLERİ

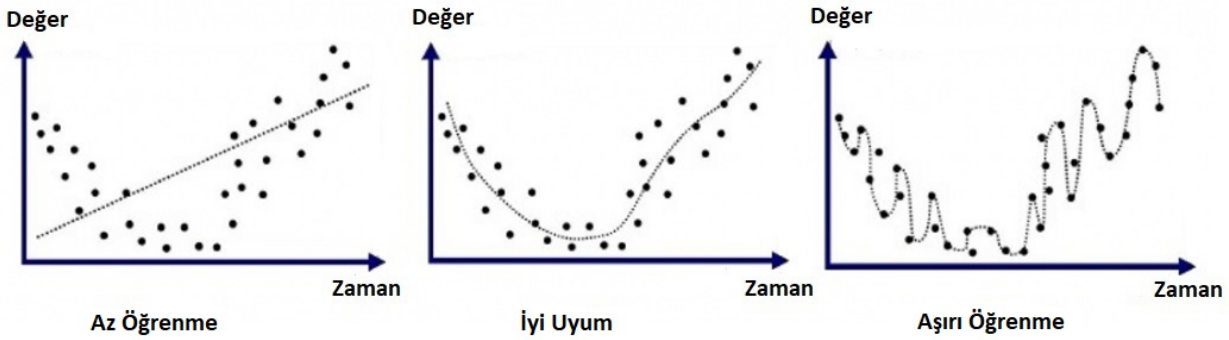
Bir modelin performansını, sadece söz konusu modeli eğitmek için kullanılan eğitim seti etkiler dersek hatalı bir söylemde bulunmuş oluruz. Bir modelin performansını etkileyen birden fazla durum vardır. Bunlardan biri modeli eğitmek için kullanılan eğitim seti olmakla beraber seçilen eğitim ve test kümelerinin boyutları da modelimizin performansını direkt etkileyecektir. Bunun dışında modelimizin *aşırı öğrenme (overfitting)* ve *az öğrenme (underfitting)* durumlarından da kaçınmasını isteriz. Performansı etkileyen bu tip durumları minimize etmek için bazı yöntemler geliştirilmiştir. Bu kısımda öncelikle model performansını direkt etkileyen unsurlardan olan aşırı öğrenme ve az öğrenme durumlarından bahsedilecek ve ardından aşırı öğrenme ve az öğrenme durumlarından kaçınmamıza imkan tanıyan yöntemlerden Çapraz Geçerleme (Cross Validation), *Birini Dışarıda Bırakarak Çapraz Geçerleme (Leave One Out Cross Validation)*, *Hold-Out Yöntemi* ve *Re-substitution Yöntemi* anlatılacaktır.

3.1 Aşırı Öğrenme

Aşırı öğrenme problemi; modelimiz, onu eğittiğimiz eğitim verisi üzerinde düşük bir hataya sahipken, eğitim kümesi dışında kalan başka bir deyişle modelin hiç görmediği (test kümesi) verilerde hatanın çok fazla olmasıdır. Bu durumu sınava girecek bir öğrencinin geçmiş sınav sorularını ezberlemesi ve sınavda ezberlediği tarzda soruların gelmemesi ve doğal olarak öğrencinin düşük not alması olarak düşünebiliriz [2].

3.2 Az Öğrenme

Az öğrenme problemini aşırı öğrenme probleminin tam tersi olarak düşünebiliriz. Eğer modelimiz eğitim kümesi üzerinde basit olarak hazırlandıysa eğitim kümesindeki hatası çok fazla olacaktır. Bununla beraber modelin hiç görmediği (test kümesi) verilerde de hata çok fazla olacaktır. Dolayısıyla gerçek değer ile tahmin edilen değer arasındaki fark açılacaktır [2].



Şekil 3.1 : Az Öğrenme, iyi uyum, aşırı öğrenme [2].

Sonuç olarak; bir modeli eğitme aşamasında aşırı öğrenme ve az öğrenme durumu göz önüne alınması gereken ve modelin performansını direkt etkileyen iki unsurdur. Az öğrenme durumu çok çabuk fark edilebilir. Çünkü az öğrenme durumu eğitim kümesinde bize çok kötü sonuçlar verecektir. Dolayısıyla az öğrenme durumu daha modeli eğitme aşamasında farkedilebilir. Fakat aşırı öğrenme durumu için aynı şey söz konusu değildir. Buradaki sorun; aşırı öğrenme durumunda eğitim kümesinde iyi sonuçlar veren modelin, hiç görmediği veriler üzerinde de (test kümesi) aynı sonuçlar verip veremeyeceğinin tespit edilebilmesidir.

Sonuç olarak; az öğrenme ile aşırı öğrenme arasında bir denge durumu olması gerekmektedir. Bu denge durumu da *Yanlılık-Varyans İkilemi (Bias-Variance Tradeoff)* olarak adlandırılmaktadır. Modelimizin bu denge durumunu sağlaması için ise çapraz doğrulama yöntemleri geliştirilmiştir.

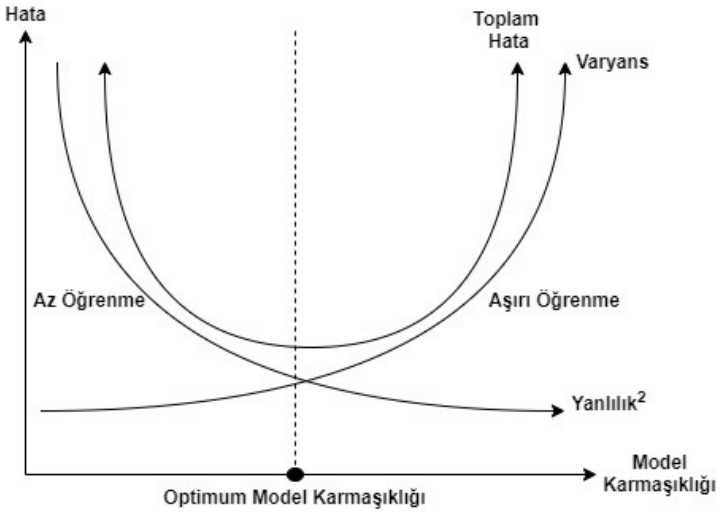
3.3 Yanlılık ve Varyans İkilemi *(Bias-Variance Tradeoff)*

Öncelikle belirtmek gerekir ki, iyi bir tahminleme performansına sahip olması istenen her gözetimli öğrenme algoritmasında amaç; *düşük yanlılık (low bias) ve düşük varyans (low variance)* durumunu elde etmektir [10].

Yanlılık ile modelin ne kadar hatalı kurulduğu ölçülür. Başka bir deyişle az öğrenme kısmında da belirtildiği gibi modelin eğitim kümesindeki hatası fazla olacaktır. Örneğin; verimizin 2. dereceden bir polinom gibi dağıldığını bir durumda modelin bu veriyi lineer bir dağılıma sahipmiş gibi görmesi örnek olarak verilebilir. Sonuç olarak yüksek yanlılığa sahip modellerde az öğrenme durumu görülmektedir.

Varyans yükseldikçe aşırı öğrenme durumuna sebep olur. Bu da modelin karmaşıklığının artması demektir. Başka bir deyişle model, eğitim kümesindeki en ufak değişimden bile etkilenebilecek durumdadır. Bu durum veri kümesi büyütülerek aşılabilir.

Kısaca, hatanın küçük olabilmesi için, uygun varsayımlarla yanlılık küçük tutulmalı ve yeterince büyük bir veri seti kullanıp modelin varyansı azaltılmalıdır.



Şekil 3.2 : Yanlılık-Varyans ikilemi.

3.4 Doğrulama Yöntemleri

Oluşturulan makine öğrenmesi modellerini değerlendirmek üzere geliştirilmiş birden çok doğrulama yöntemi vardır.

3.4.1 Çapraz doğrulama *(Cross validation)*

Aşırı öğrenme problemini aşmak için çapraz doğrulama yöntemleri kullanılabilir. Çapraz doğrulama yönteminde tüm veri iki ayrı alt kümeye ayrılır; eğitim kümesi ve test kümesi. Burada eğitim kümesi modeli eğitmek için kullanılırken, test kümesi ise eğitilen modelin performansını tespit etmek için kullanılmaktadır.

3.4.1.1 k-katlı çapraz doğrulama *(k-fold cross validation (KFCV))*

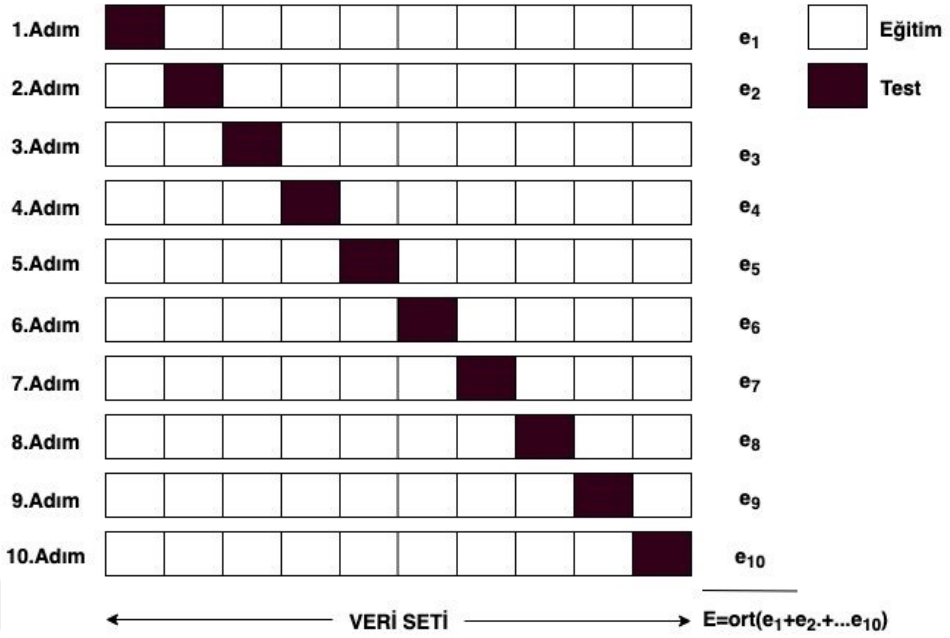
Gözetimli öğrenme algoritmalarında bir öğrenme modelinin oluşturulması kadar bu öğrenme modelinin değerlendirilmesi de büyük önem taşımaktadır. Bir modelin değerlendirilmesi demek, o modelin bir t zamanında ne kadar doğru tahminde bulunacağını gözlemlemektir.

Gözetimli öğrenme algoritmalarında bir öğrenme modeli oluşturulurken sadece eğitim veri kümesini kullanmak modelimizi aşırı öğrenme durumuna maruz bırakabilir. Aşırı öğrenme durumundan kaçınmak için önerilen yöntemlerden biri k-katlı Çapraz Doğrulama (*k-fold Cross Validation*) yöntemidir. Bu yöntemde aşırı öğrenme durumundan kaçınmak için öğrenme modeli oluşturulması aşamasında test kümesi kullanılmaktadır. Yöntem elimizdeki $D = \{x_1, x_2, ..., x_n\}$ veri kümesinin k tane alt kümeye bölünmesi ile başlar. k değeri veri kümemizi kaç alt kümeye böleceğimizi temsil eden bir parametredir. $D = \{x_1, x_2, ..., x_n\}$ veri kümesi için seçilen herhangi bir k parametresi ile bu yöntem şu şekilde çalışır:

- n tane veri içeren D veri kümesi k tane alt kümeye bölünür.
- İlk iterasyonda birinci alt küme test kümesi olarak ayrılırken, kalan $(k-1)$ tane alt küme eğitim kümesi olarak ayrılır.
- Her iterasyonda bir alt küme test kümesi ve kalan $(k-1)$ alt küme eğitim kümesi olacak şekilde tekrarlanır.
- Bu işlem k defa tekrarlanır. Her tekrarda, y_i tahmin edilen ve x_i ise gerçek değeri temsil etmek üzere, $e_i = |y_i - x_i|^2$ hatası hesaplanır.
- Elde edilen k tane test kümesine ait hataların ortalaması alınarak yöntemin toplam hatası hesaplanır. [12]

3.4.1.2 Birini dışarıda bırakarak çapraz doğrulama *(Leave one out cross validation (LOOCV))*

KFCV Yöntemi'ne oldukça benzeyen LOOCV Yöntemi, KFCV Yöntemi'nin özel bir hali olarak düşünülebilir. Tüm veri kümesinde n tane veri olduğu düşünülürse, bu yöntem; tüm veri kümesinden rastgele seçilen bir tane veri ile model test edelirken, kalan $(n-1)$ tane veri ile modeli eğitilir. Başka bir deyişle, veri kümesindeki her bir veri test verisi oluncaya kadar yöntem devam ettirilir. Yani burada $k = n$ diyebiliriz.



Şekil 3.3 : k=10 için k-katmanlı çapraz doğrulama.

3.4.2 Hold-Out yöntemi

Hold-Out Yöntemi' de aynı KFCV gibi tüm veri kümesini eğitim ve test olmak üzere iki ayrı ve ayrık alt kümelere ayırma üzerine kurulu bir yöntemdir. Fakat KFCV' den farklı olarak bu yöntemde veri kümesi sadece bir kere eğitim ve test kümesi olarak ayrılır. Bu ayrımın sadece bir kez yapılmasından dolayı hold-out yöntemi ile kurulacak olan modelin performansı yapılan bu ayrıma oldukça bağlıdır. KFCV' de böyle bir bağımlılık söz konusu değildir. Çünkü; *k* tane farklı eğitim ve test kümesi olduğundan KFCV ile kurulacak olan modelin performansını eğitim ve test kümesi ayrımına bağlılığı oldukça azdır.

3.4.3 Re-substitution yöntemi

Bu yöntemde tüm veri seti modeli eğitmek için kullanılır. Modeli eğitmek için kullanılan aynı veri seti ile de model test edilir. Fakat bu yöntem ile eğitilen modellerde aşırı öğrenme durumu görülebilir.

3.4.4 Hangi doğrulama yöntemi nerede kullanılmalıdır?

Çok büyük sayıda veri içeren sistemlerde LOOCV Yöntemi' nin kullanımı önerilmez. Çünkü; LOOCV Yöntemi'nde model sistemdeki veri sayısı *n* kadar eğitileceğinden *n* değeri büyüdükçe bu yöntem pahalı olacaktır. Dolayısıyla daha az sayıda veri içeren sistemlerde LOOCV Yöntemi'ni kullanmak daha mantıklı olacaktır. Hold-out yöntemi tek bir eğitim-test bölmesi yaptığından büyük veri setleri için KFCV ile karışılaştırıldığında zaman açısından bir avantaj sağlayabilir. Fakat az öğrenme durumu daha yaygın olarak görülebilir.

4. AYRIK DEĞİŞKENLERLE MODEL PERFORMANSI ÖLÇME

4.1 Model Performans Ölçüleri

Bir model kullanılarak yapılan sınıflandırma, kümeleme işlemleri sonucu kullanılan modelin ne kadar başarılı olduğunu ölçmek isteriz. Bunlardan en çok kullanılanı hata matrisi ya da karışıklık matrisi *(confusion matrix)* olarak da adlandırılıan bir matris yardımıyla modelin başarısını hesaplamaktır.

Çizelge 4.1 : İki olası sınıflı model için hata matrisi.

		Modelin Tahmin Ettiği Sınıf	
		Pozitif	Negatif
Gerçek Sınıf	Pozitif	TP_P	FN_{PN}
	Negatif	FP_{NP}	TN_N

Çizelge (4.1)' de model tarafından üretilen *(predicted)* ve gerçek *(actual)* değerleri barındıran hata matrisi verilmiştir. Çizelge (4.1)' deki bazı kavramlara değinelim;

- TP (True Positive, Doğru Pozitif): Gerçekte *Pozitif* olan verilerin kaç tanesini modelin *Pozitif* olarak tahmin ettiğinin sayısıdır.
- FN (False Negatif, Yanlış Negatif): Gerçekte *Pozitif* olan verilerden kaç tanesini modelin *Negatif* olarak tahmin ettiğinin sayısıdır.
- FP (False Positive, Yanlış Pozitif): Gerçekte *Negatif* olan verilerin kaç tanesini modelin *Pozitif* olarak tahmin ettiğinin sayısıdır.
- TN (True Negative, Doğru Negatif): Gerçekte *Negatif* olan verilerin kaç tanesini modelin *Negatif* olarak tahmin ettiğinin sayısıdır. .

Aşağıda Çizelge (6.1) kullanılarak *doğruluk (accuracy), kesinlik (precision), duyarlılık (recall)* kavramları ve formülleri verilmiştir.

- **Doğruluk**: Gerçek sınıflarla modelin tahmin ettiği sınıfların ne oranda örtüştüğünü söyler.

$$\text{Doğruluk} = \frac{TP+TN}{TP+FP+FN+TN} \tag{4.1}$$

Bir sınıflandırma modelinin sınıflandırma başarısını ölçmek için genelde sadece doğruluk değerine bakmanın yeterli olacağı düşünülür. Fakat bu doğru değildir. Örneğin; dengesiz veri seti *(imbalanced data set)* olarak adlandırılan yani sınıflar arası dağılımın düzensiz olduğu veri setlerinde başarı kriteri olarak sadece doğruluk değerine bakmak bizi yanıltabilir. Bu duruma örnek verebilmek adına aşağıda verilen Çizelge (4.2)' yi inceleyelim.

Çizelge 4.2 : Sınıf dağılımı dengesiz olan bir veri seti için hata matrisi.

		Modelin Tahmin Ettiği Sınıf	
		Siyah	Beyaz
Gerçek Sınıf	Siyah	90	210
	Beyaz	140	9560

Çizelge (4.2)' deki hata matrisindeki doğruluk=%96.40 olarak görülmektedir. Sadece bu değere bakarak modelimizin mükemmel bir başarıyla sınıflandırma yaptığı düşünülebilir. Fakat duyarlılık değerine bakacak olursak duyarlılık=%30 olarak görülecektir. Yani bu durumu şöyle açıklayabiliriz. Modelimizin genel başarısı her ne kadar %96.40 gibi yüksek bir oran ise *Siyah* sınıflı verileri tahmin etme başarısı ise %30 olarak görülmektedir. Yani modelimiz genel başarı değerinin aksine *Siyah* sınıflı verileri %30 gibi düşük bir başarı değeri ile sınıflandırmaktadır. Bu kabul edilebilir bir durum değildir. Daha çarpıcı bir örnek verecek olursak; Bir grup hastayı düşünelim ve bu hasta grubundaki insanların kanser olup olomadığını tahmin etmeye çalışalım. Çizelge (4.2)' deki *Siyah* sınıfını *Kanser*, *Beyaz* sınıfını ise *Kanser Değil* olarak düşünelim. Benzer şekilde yine modelimizin genel başarısı %96.40 iken modelimizin *Kanser* olan hastaları doğru sınıflandırabilme oranı %30 olacaktır. Bu durum bize modelin Kanser olan hastaları çok düşük bir oranda doğru sınıflandırabildiğini söyler ve bu kabul edilebilir bir durum değildir.

- **Kesinlik**: Doğru sınıflandırılan pozitif sınıfların pozitif olarak tahmin edilen sınıfların toplamına oranıdır. Pozitif Öngörü Değeri *(Positive Predicted Value)* de denmektedir.

$$\text{Kesinlik} = \frac{TP}{TP+FP} \tag{4.2}$$

Doğru sınıflandırılan negatif sınıfların negatif olarak tahmin edilen sınıflara oranı da Negatif Öngörü Değeri *(Negative Predicted Value)* olarak adlandırılır.

  - **Pozitif Olabilirlik Oranı:** Bir sınıflandırma sonucunun pozitif sınıfların varlığında pozitif çıkma olasılığının, negatif sınıfların varlığında pozitif çıkma olasılığına oranıdır.
  - **Negatif Olabilirlik Oranı:** Bir sınıflandırma sonucunun negatif sınıfların varlığında negatif çıkma olasılığının, pozitif sınıfların varlığında negatif çıkma olasılığına oranıdır.

Pozitif olabilirlik oranının yüksek olması gerçekte pozitif olan sınıfların o denli iyi sınıflandırıldığı, negatif olabilirlik oranının yüksek olması gerçekte negatif olan sınıfların o denli iyi sınıflandırıldığı anlamına gelir [13].

Yukarıda verilen kanser örneğinden yola çıkarsak, kesinlik değeri bir kişiye kanser demeden önce iyi düşünüp taşınmayı amaçlıyor. Yani biz kanser olan hastaları doğru tahmin etmek adına herkese kanser dersek kesinlik değeri $300/10000 = 0.03$ gibi kötü bir değer çıkacaktır. Diğer taraftan bir kişi gerçekten hasta ve biz ona hasta dediysek kesinlik değeri 1 olacaktır. Fakat bu durumda da kalan hastaları tespit edemediğimiz için FP değeri yüksek olacaktır.

- **Duyarlılık**: Modelin pozitif etiketli sınıfları doğru tahmin edebilme başarısıdır.

$$\text{Duyarlılık} = \frac{TP}{TP+FN} \tag{4.3}$$

Bazı anomali durumlarını doğru tespit etmek anomali durumlarının dışındaki durumlar için yanlış tahminde bulunmaktan daha önemlidir. Yukarıda verilen kanser örneğinde olduğu gibi kanser olan bir kişiyi doğru tespit edebilmek kanser olmayan biri için yanlış bir tespitte bulunmaktan daha önemlidir. İşte bu

tip durumlarda duyarlılık metriğinin önemi daha iyi anlaşılmaktadır. Anomali durumlarını doğru tespit etmeye çalışırken *FN* değeri yükselecektir.

Görüldüğü üzere en az doğruluk kadar önemli iki metrik olan duyarlılık ve kesinlik arasında bir takas *(trade-off)* var. Bu takası doğru bir şekilde ifade edebilmek ve bir modelin başarısını daha sağlıklı hesaplayabilmek için *F-Ölçütü (F-Measure)* kullanılmaktadır.

4.2 F-Ölçütü

Bazı durumlarda duyarlılık ve kesinlik de anlamlı sonuçlar vermeyebilir. Bu durumda bu iki ölçütü bir arada kullananan *F-Ölçütü* yardımıyla daha doğru sonuçlar elde edebiliriz. F-Ölçütü duyarlılık ve kesinlik değerlerinin harmonik ortalamasıdır [12]. İyi bir F-Ölçütü değeri, düşük FP ve düşük FN değeri üretir. Bu da demektir ki minimum yanlış alarm üreterek maksimum seviyede verileri ayrıştırabilmek demek. F-Ölçütü' nün genel ifadesi

$$F_\alpha = (1+\alpha)\frac{Kesinlik.Duyarlılık}{\alpha.Kesinlik + Duyarlılık} \tag{4.4}$$

şeklinde ifade edilir. Burada α değeri duyarlılık ve kesinlik arasındaki önem derecesine göre değişkenlik göstermektedir.

Eğer,

$\alpha = 1 \implies$ duyarlılık ve kesinlik eşit öneme sahiptir.

$\alpha = 2 \implies$ duyarlılık kesinlikten iki kat daha fazla önemlidir.

$\alpha = 0.5 \implies$ kesinlik duyarlılıktan iki kat daha fazla önemlidir.

yorumları yapılabilir. Buradaki önem derecesini açıklamak için şu yorumu yapabiliriz; bazı durumlarda Çizelge (4.1)' de verilen FN değeri FP değerinden daha kritik olabilir. Örneğin; kanser hastası olan birini doğru sınıflandırmayıp O' nun ölümüne sebep olmaktansa kanser olmayan hastaları yanlış sınıflandırmak daha kabul edilebilirdir. Dolayısıyla kesinlik ve duyarlılık arasındaki önem çekişmesi bu şekilde açıklanabilir.

Eğer modelimiz ikiden fazla farklı sınıf içeriyorsa bu durumda örnek hata matrisi aşağıdadır.

Çizelge 4.3 : Üç sınıflı bir model için örnek hata matrisi.

		Modelin Tahmin Ettiği Sınf		
		A	**B**	**C**
Gerçek	**A**	TA	FB $_{AB}$	FC$_{AC}$
	B	FA$_{BA}$	TB	FC$_{BC}$
Sınıf	**C**	FA$_{CA}$	FB$_{CB}$	TC

Yukarıda verilen üç sınıflı hata matrisinde doğruluk oranı

$$\text{Doğruluk} = \frac{TA+TB+TC}{FA_{BA}+FB_{AB}+FC_{CA}+FB_{CB}+FC_{BC}} \tag{4.5}$$

olarak hesaplanabilir. Duyarlılık, kesinlik ve belirleyicilıik de her bir sınıf için ayrı ayrı hesaplanır. A,B ve C sınıfları için duyarlılık,

$$\begin{aligned} \text{Duyarlılık}_A &= \frac{TA}{TA+FB_{AB}+FC_{AC}} \\ \text{Duyarlılık}_B &= \frac{TB}{TB+FA_{BA}+FC_{BC}} \\ \text{Duyarlılık}_C &= \frac{TC}{TC+FA_{CA}+FB_{CB}} \end{aligned} \tag{4.6}$$

şeklindedir. Benzer şekilde her bir sınıf için kesinlik ve belirleyicilik değerleri hesaplanabilir. Her bir sınıf için bulunan duyarlılık ve kesinlik değerleriyle F-Öçütü bulunur. Modelin F-Ölçütü, her bir sınıf için hesaplanan F-Ölçütlerinin ortalamasıdır.

Bu metriklerin yanı sıra iki model arasında bir seçim yapılmak isteniyorsa bu metriklere ek olarak iki farklı modelin birbiri ile uyumunu nicel bir çıktı vererek inceleyen *Cohen Kappa Skor (Cohen's Kappa Score)* istatistiğini incelemek gerekir.

4.3 Cohen Kappa Skoru / İstatistiği *(Cohen's Kappa Score)*

Kappa skoru olarak adlandırlan bu metrik kategorik veriler arasındaki uyumu ölçer. Uyumun değerlendirdiği değişken kategorik olduğundan bu test parametrik olmayan test sınıfına girer. Kappa skoru iki sınıflı veriler için kullanılabiliyorken daha fazla sınıfa sahip veriler için *Fleiss Kappa Skoru / İstatistiği (Fleiss Kappa Score)* kullanılır.

Kappa skoru aradaki uyumun şans eseri olabileceğini hesaba katarak bir değerlendime yapar. Bu durum kappa skorunu doğruluk oranına göre daha iyi bir performans metriği yapmaktadır.

Kappa Skoru hesaplanırken iki farklı olasılık hesabı yapılır. Bunlar $Pr(a)$ ve $Pr(e)$ olasılıklarıdır. $Pr(a)$ olasılığı, aradaki uyumların toplam orantısı iken $Pr(e)$ olasılığı, söz konusu uyumun şansa bağlı olması durumudur. Bu bilgiler ışığında Cohen Kappa Skoru,

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} \tag{4.7}$$

formülü ile hesaplanır [14].

Çizelge 4.4 : Temsili bir hata matrisi.

	c_1	c_2
c_1	a	c
c_2	b	d

c_1 ve c_2 olacak şekilde iki kategorik sınıf değişkenine sahip Çizelge (6.4)' deki temsili bir hata matrisi için $Pr(e)$ hesabı

$$Pr(e) = \frac{a+b}{a+b+c+d} \cdot \frac{a+c}{a+b+c+d} + \frac{c+d}{a+b+c+d} \cdot \frac{b+d}{a+b+c+d} \tag{4.8}$$

şeklindedir.

Kappa skoru K ile gösterilir ve $[-1,1]$ aralığında değerler alır. $K = 1$ olması durumu mükemmel uyum olarak adlandırılır. $K < 0$ durumunun güvenirlilik açısından bir anlamı yok iken, $K = 0$ uyumun şansa bağlı olduğunu ve $K \geq 0.4$ olması durumu genelde makul bir uyum olduğunu ifade eder.

Aşağıda 1977 yılında *Landis, J.* tarafından ortaya atılan Kappa skorlarının Kappa skor aralıkları ve bu aralıklara karşılık gelen yorumlar verilmiştir.

Çizelge 4.5 : Kappa skorları ve yorumları [1].

Kappa Skoru	**Yorum**
0.81-1.00	Mükemmel Uyum
0.61-0.80	Önemli Derecede Uyum
0.41-0.60	İyi Derecede Uyum
0.21-0.40	Orta Derecede Uyum
0-0.20	Kötü Uyum
0	Şansa Bağlı Uyum

Örnek olarak, bir model düşünelim. Bu modelin *siyah* ve *beyaz* şeklinde olası iki farklı sınıflı bir veri seti üzerinde yaptığı tahminlerin sonucunda oluşan hata matrisi

Çizelge 4.6 : Örnek bir modele ait hata matrisi.

		Tahmin Edilen	
		Siyah	Beyaz
Gerçek Sınıf	Siyah	6	1
	Beyaz	1	2

olsun.

Toplamda 10 tane tahminden 6' sında *siyah* üstünde bir anlaşma vardır . Bununla beraber 10 tahminden 2 tanesinde ise *beyaz* üstünde bir anlaşma vardır. Dolayısıyla bu modelin uyumu yani $P(a) = 8/10 = 0.80$ olarak bulunur. Öte yandan uyumun şansa bağlı gerçekleşme durumunu yani $Pr(e)$' yi hesaplayalım.

- Model, $7(7/10 = 0.70)$ tane rengi *siyah* olarak tahmin ederken, $3(3/10 = 0.30)$ rengi de *beyaz* olarak tahmin etmiştir.
- Gerçekte, $7(7/10 = 0.70)$ tane *siyah* varken $3(3/10 = 0.30)$ tane de *beyaz* vardır.
- Bu bilgiler ışığında modelin rastgele olarak *siyah* ve *beyaz* tahmininde bulunma olasılıkları sırasıyla $0.70 * 0.70 = 0.49$ ve $0.30 * 0.30 = 0.09$ olarak bulunur.
- O halde bu modelin toplam şansa bağlı uyum olasılığı olan $Pr(e) = 0.49 + 0.09 = 0.58$ olarak bulunur.

Bulunan bu değerleri kullanarak Kappa Skoru

$$\kappa = \frac{0.80 - 0.58}{1 - 0.58} = 0.52 \tag{4.9}$$

olarak bulunur.

5. MAKİNE ÖĞRENMESİ ALGORİTMALARI

5.1 Makine Öğrenmesi

Makine Öğrenmesi ya da Yapay Öğrenme, bir sistemin örnek veri ve deneyimler vasıtasıyla oluşturulan bir hesaplamalı model ile ileriye dönük tahminler yapması ya da eldeki verinin yapısı ve dağılımı hakkında istatistiksel çıkarımlar yapmasıdır. Genel olarak iki alt başlık altında incelenirler. Bunlardan biri Gözetimli Öğrenme (Supervised Learnnig) diğeri ise Gözetimsiz Öğrenme (Unsupervised Learning)' dir.

5.1.1 Gözetimli öğrenme

Gözetimli öğrenme, veri kümesindeki her bir x özniteliğinin i. niteliği x_i ile ilgili sınıf *(hedef)* değişken Y_i arasında $Y = f(x)$ ilişkisi kuran bir makine öğrenmesi yöntemidir. Gözetimli öğrenme yöntemlerinde amaç; $Y_i = f(x_i)$ fonksiyonunun her yeni x_i değeri için Y_i çıkış değerini tahmin etmektir.

Gözetimli öğrenmede tahmin etme işlemi test verisi *(test set)* üzerinden gerçekleşir. Eğitim verisi ile öğrenilen veriler artık bir etikete (*(labeled)*) sahip olur ve bu veriler üzeriden bir model oluşturulur. Oluşturulan model temel alınarak eğitim verisi dışında kalan (etiketli olmayan (*unlabeled*)) yani test kümesi *(test set)* ile x_i değerleri için bir etiket yani bir Y_i değeri tahmin edilmeye çalışılır. Gözetimli Öğrenme algoritmalarına *K-en yakın komşu (KNN), Logistik Regresyon, Destek Vektör Makineleri (SVM), Karar Ağaçları, Naive Bayes Sınıflandırıcı* gibi algoritmalar gözetimli öğrenme algoritmalarındandır ve bu tez kapsamında incelenecekleridir.

5.1.2 Gözetimsiz öğrenme

Gözetimli öğrenmede bir x_i verisinin Y_i çıkış değerini öğrenmek için eğitim verileri ile kurulan bir modele ihtiyaç vardı. Gözetimsiz öğrenmede ise herhangi bir etiketleme işlemi yapılmadığından bir eğitim verisine ihtiyaç yoktur. Dolayısıyla hangi verinin hangi sınıfta olduğu konusunda da herhangi bir bilgiye sahip

değiliz. Sonuç olarak gözetimsiz öğrenme, etiketli olmayan verilerin arasındaki ilişkinin belli benzerlik ölçütleri kullanarak öğrenilmesidir. Gözetimsiz öğrenmeye Kümeleme Algoritmaları örnek olarak gösterilebilir. Bunlardan en yaygın olarak kullanılanı K-Means algoritması olup bu tezin kapsamında bu algoritma incelenecektir.

5.2 K-Ortalamalar Yöntemi *(K-Means)*

K-Means, 1967 yılında J.B. MacQuenn tarafından öne sürülen bir gözetimsiz öğrenme algoritmasıdır [15].

K-Means, bir veri kümesinde verilen n adet veriyi, başlangıçta tanımlanma zorunluluğu olan ve $K \leq n$ olacak şekilde K adet kümeye bölmeyi amaçlar. K tane küme $c_K \in S_K$ olacak şekilde bir c_K merkez noktası ile temsil edilir [16]. Verilerin K adet kümeye bölünmesinin ardından, kümeler içi benzerliğin artırılması ve kümeler arası benzerliğin azaltılması gerekir. Bu durum *Kümeler Arası Toplam Karesel Hata (Within Cluster Sum of Square (WCSS))* olarak adlandırılır.

WCSS değeri, küme içi değişim e_j^2 olarak tanımlanan değerlerin toplamından oluşur. Küme içi değişim; küme merkezi ile küme merkezi dışında kalan verilerin farkının karelerinin toplamıdır.

$$e_j^2 = \sum_{j=1}^{K} \sum_{i=1}^{n} (x_i - c_j)^2 \tag{5.1}$$

O halde K tane küme için WCSS değeri K tane kümenin küme içi değişimlerinin toplamıdır.

$$WCSS = \sum_{j=1}^{K} e_j^2 \tag{5.2}$$

Özet olarak; kümeleme işleminin sonuna gelindiğinde K tane kümenin karesel hataları toplamı yani WCSS değeri minimum olmalıdır.

K-Means algoritması en çok kullanılan kümeleme algoritması olmasına rağmen bazı dezavantajları vardır. Bunlardan en önemlisi ise; başlangıçta algoritmaya verilerin kaç tane kümeye ayrılmasını söylemektir. Yani K değerinin belirlenmesidir. K değerinin belirlenme sürecini daha sonra bu başlık altında inceleyeceğiz.

Algoritma:

$\forall x_i \in \mathbb{R}^n$ olacak şekilde $D = \{x_1, x_2, ..., x_n\}$ veri kümesi verilsin.

1. $\forall c_i \in \mathbb{R}^n$ olacak şekilde K tane merkez noktası $(c_1, c_2, ..., c_K)$ seçilir. Bu seçim rastgele olacak şekilde gerçekleştirilir.
2. $\forall x_i \in D$ için $\sum_{j=1}^{K} \sum_{i=1}^{n} d(x_i, c_j)$ uzaklığı hesaplanır. Uzaklık hesabında kullanılan $d(x_i, c_j)$ uzaklığı başka uzaklık metrikleri ile hesaplanabileceği gibi aşağıdaki Öklid Uzaklığı (*Euclidean Distance*) yardımı ile de hesaplanabilir:

$$d(x_i, c_j) = (|x_{i1} - c_{j1}|^2 + |x_{i2} - c_{j1}|^2 + \cdots + |x_{in} - c_{jn}|^2)^{\frac{1}{2}}$$

   $S_i \cap S_j = \emptyset, i \neq j$ olacak şekilde her bir $S_j (j = 1, 2, \ldots, K), c_j$ merkezi ile temsil edilen bir küme olsun. $argmind(x_i, c_j)$ ve $x_i \in S_j$ olacak şekilde her bir c_j merkez noktasına en yakın x_i noktaları belirlenir ve S_j kümesine dahil edilir. Bu şekilde n tane veri K tane S_j kümesine yerleştirilmiş olur.
3. Her S_j kümesi için $yenic_j = \frac{1}{|S_j|} \cdot \sum_{x_i \in S_j} x_i$ hesaplanır.
4. Eğer:
   (a) Her j değeri için $|yenic_j - c_j| < \varepsilon$ ise kümeleme işlemi sonlandırılır
   (b) En az bir j değeri için $|yenic_j - c_j| > \varepsilon$ ise 2. adıma dönülür ve adımlar iteratif olarak tekrarlanır [17].

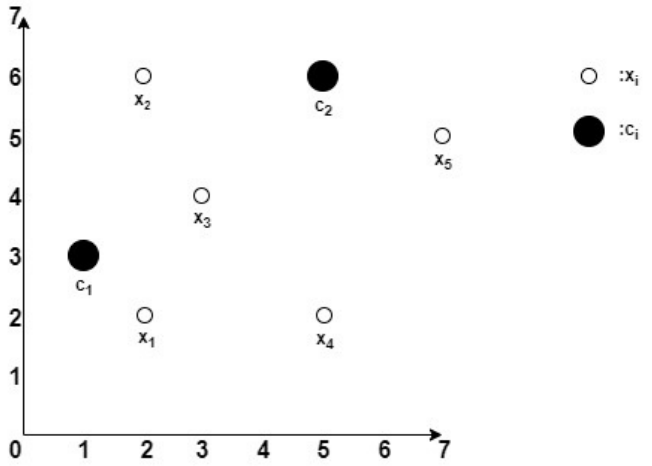
Aşağıdaki örnek veri seti için K-Means algoritmasını uygulayalım:

Çizelge 5.1 : 2 boyutlu düzlemde örnek bir veri seti.

Veri	X	Y
x_1	1	3
x_2	2	6
x_3	3	5
x_4	4	3
x_5	5	5

- **1.Adım:**

Öncelikle threshold değerimiz olan ε belirleyelim. $\varepsilon = 3$ olsun.



Şekil 5.1 : Rastgele seçilen merkez noktalarının ve verilerin görünümü *(K = 2)*.

$K = 2$ olarak seçildiğini varsayalım. Rastgele seçilen merkez noktaları $c_1 = (1,3)$ ve $c_2 = (5,6)$ ile beraber verilerin görünümü aşağıdaki gibi olacaktır.

- **2.Adım:**

  Şekil (5.1)' den yararlanarak, her c_i merkez noktası ile veriler arasındaki uzaklığı belirleyelim. Belirlenen uzaklığa göre en yakın olan verileri c_i merkezli S_i kümesine atayalım. Uzaklığı belirleme noktasında öklid uzaklığını kullanalım.

  Başlangıçta rastgele seçilen c_1 küme merkezine olan uzaklıklar:

  Çizelge 5.2 : c_1 küme merkezine uzaklıklar.

	x_1	x_2	x_3	x_4	x_5
c_1	$\sqrt{2}$	$\sqrt{10}$	$\sqrt{5}$	$\sqrt{17}$	$2\sqrt{10}$

  Başlangıçta rastgele seçilen c_2 küme merkezine olan uzaklıklar:

  Çizelge 5.3 : c_2 küme merkezine uzaklıklar.

	x_1	x_2	x_3	x_4	x_5
c_2	5	3	$2\sqrt{2}$	4	$\sqrt{5}$

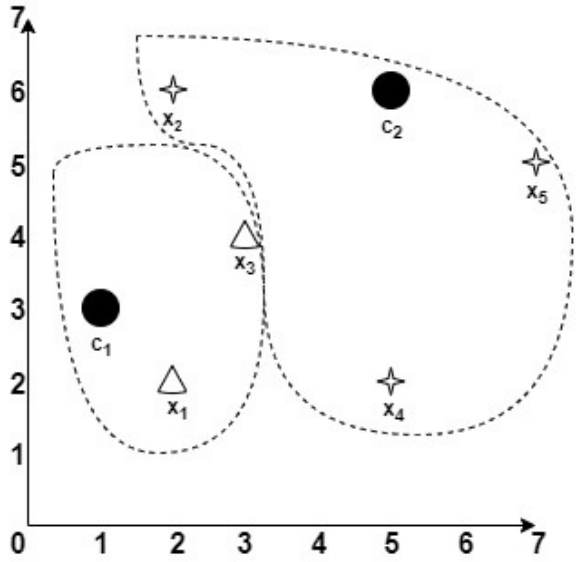
  Bu durumda S_1 ve S_2 kümeleri; $S_1 = \{x_1, x_3\}$ ve $S_2 = \{x_2, x_4, x_5\}$ olur.

- **3.Adım:**

  Bu adımda, 2.adımda yapılan kümeleme işleminden sonra S_1 ve S_2 kümeleri için yeni merkez noktalarını $(yenic_1)$ ve $(yenic_2)$ hesaplamamız gerekiyor.

  $yenic_1 = \{\frac{x_{11}+x_{12}}{2}, \frac{x_{21}+x_{22}}{2}\} = \{\frac{2+3}{2}, \frac{2+4}{2}\} = \{2.5, 3\}$

  $yenic_2 = \{\frac{x_{21}+x_{41}+x_{51}}{3}, \frac{x_{22}+x_{42}+x_{52}}{3}\} = \{\frac{2+5+7}{3}, \frac{6+2+5}{3}\} = \{4.7, 4.3\}$



Şekil 5.2 : S_1 ve S_2 kümelerinin görünümü.

- **4.Adım:**

  $|yenic_j - c_j| < \varepsilon = 2,9$ olduğundan başta belirlediğimiz threshold değerine göre kümeleme işlemimiz bitmiş oldu. Eşik değeri *(threshold)* değeri, epsilon sayısının doğası gereği çok küçük bir sayı olmalıdır. Burada örneğe uygun bir değer seçilmiştir.

5.2.1 Optimum K değeri seçimi

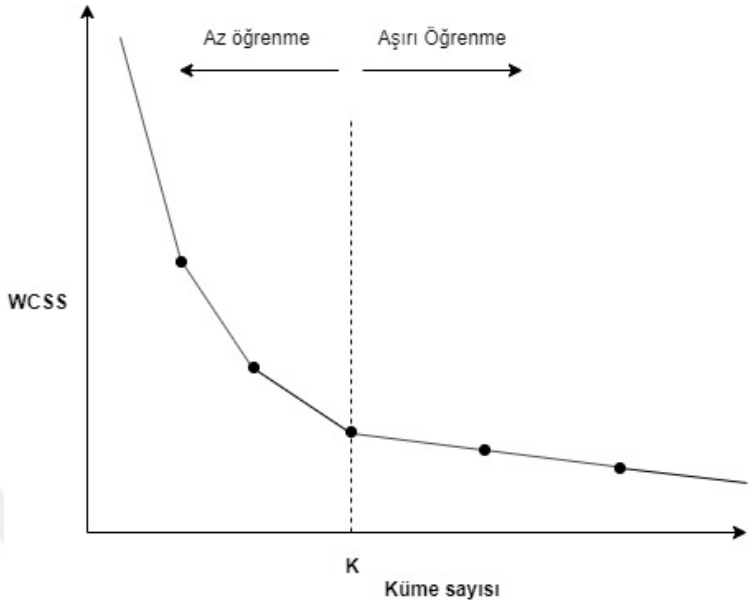
Daha önce K-Means algoritmasının dezavantajlarından biri olarak K değeri seçimi zorunluluğundan bahsetmiştik. K değerini seçmek için bir çok yöntem vardır. Biz burada *Dirsek Yöntemi (The Elbow Method)*'ni inceleyeceğiz.

Her bir $x_i \in \mathbb{R}^p, (i = 1,...,n)$ olmak üzere $D = \{x_1, x_2, ..., x_n\}$ veri kümesi verilmiş olsun. K ise n tane veriyi bölmek istediğimiz küme sayısını temsil etsin. K-Means kümeleme algoritmasında amaç; her bir S_j kümesinin c_j küme merkezi ile S_j'nin c_j'den farklı noktaları arasındaki uzaklığını minimize etmek olduğunu belirtmiştik. ve şu şekilde ifade etmiştik:

$$WCSS = \sum_{j=1}^{K} \sum_{x_i \in S_j} (x_i - c_j)^2 \tag{5.3}$$

Şekil (5.3)'teki grafikte WCSS değeri ile K küme sayısı arasındaki ilişki verilmiştir. Grafiği yorumlayacak olursak; WCSS değeri düştükçe $x_i \in S_j$ olacak şekilde her bir x_i, küme merkezi olan $c_j \in S_j$'ye daha da yaklaşacaktır. Dolayısıyla bu da daha

iyi bir kümeleme demektir. Fakat grafikte de görüldüğü gibi az öğrenme ve aşırı öğrenme durumlarına dikkat etmek gerekir.



Şekil 5.3 : WCSS ve K küme sayısı arasındaki ilişki.

Burada aşırı öğrenme durumunu şöyle açıklayabiliriz; her bir $x_i \in D$ noktası bir küme olarak düşünülebilir. Bundan dolayı her bir x_i aynı zamanda birer küme merkezi olacaktır. Bu durum Denklem (5.2)'deki WCSS değerini sıfır yapacaktır. Fakat bu durum kümeleme mantığı ile örtüşmemektedir. Bizim amacımız birbirine benzeyen verileri sınırlı sayıda K tane kümeye ayırmaktı.

Şekil (5.3)'teki yöntem kullanıcıya K küme sayısını kesin bir şekilde vermez. Bu yöntem biraz da grafik üzerinde yorum yapmaya dayalı bir yöntemdir. Şekil (5.3)'teki grafikte görüldüğü gibi WCSS değeri K noktasına kadar hızlı bir şekilde düşüyor. K noktasından sonra ise daha yatay bir şekilde ierliyor. Yani K noktasından sonraki değerler WCSS değeri üzerinde ciddi bir azalmaya sebep olmadığı gibi modelin yorumlanabilirliğini de azaltıyor. Dolayısıyla bu yorumlardan yola çıkarak, söz konusu veri kümesi için optimum küme sayısı K olarak belirlenebilir.

5.3 K-En Yakın Komşu *(K-Nearest Neighbor (KNN))*

1951 yılında Fix, E. ve Hodges, L. tarafından tanıtılan K-NN algoritması yaygın olarak kullanılan bir sınıflandırma algoritmasıdır [5]. Bu algoritmada

sınıflandırılmak istenen verinin sınıfı, seçilen K değerine bağlı olarak belli bir yarıçap içinde aynı sınıfa ait sayıca çok x_i verilerinin sınıfı olarak belirlenir. [18]

K-NN algortiması bir gözetimli öğrenme algoritmasıdır. Bu nedenle algoritmayı kullanarak bir verinin sınıfını tahminlemeden önce bir eğitim verisi ile algoritmayı eğitmek gerekir. Genel bir ifade ile orijinal veri kümesi, eğitim kümesi ve test kümesi olmak üzere iki ayrı ve ayrık alt kümeye ayrılır. Sınıfı belirlenmek istenen veri, eğitim kümesinin bir elemanı olmamalıdır. Böyle bir verinin sınıfı kullanıcı tarafından belirlenen komşu sayısı (K) değerine bağlıdır. Ayrıca K-NN algoritması ile yapılan sınıflandırmanın başarısı yine seçilen K değerine bağlıdır. [19]

$D = \{x_1, x_2, ..., x_n\}$ veri kümesi ve her $x_i \in \mathbb{R}^p, (i = 1, 2, .., n)$ olsun. Eğitim ve test kümeleri, $D_{egitim} \cap D_{test} = \emptyset$ olacak şekilde oluşturulmalıdır.

Her bir $c^{'}, x_i^{'} \in D_{egitim}$ verisinin sınıfını belirtmek üzere eğitim veri kümesi $D_{egitim} = \{x_i^{'}, c^{'}\}$ olarak yazılsın. Öte yandan c^*, sınıfı belirlenmek istenen $x_i^* \in D_{test}$ verisinin belirlenmek istenen sınıfını temsil etmek üzere $x^* \in D_{test}$ verisinin c^* sınıfı aşağıdaki algoritma ile tespit edilir.

Algoritma:

1. Sınıfı belirlenmek istenen x^* verisine en yakın uzaklıkta bulunan K tane $x_i^{'} \in D_{egitim}$ komşu verileri belirlenir. Uzaklık şu şekilde hesaplanabilir fakat başka metrikler de kullanılabilir:

$$d(x^*, x_i^{'}) = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{p} \|x_{1j}^* - x_{ij}^{'}\|^2} \tag{5.4}$$

2. İlk aşamada x^* verisine en yakın K tane komşu belirlendi. Şimdi bu K en yakın komşu arasında sayısal olarak en fazla olan $c_i^{'}$ sınıfını belirleyip, x^* verimizin sınıfı olarak atamalıyız. Bu adımı daha basit bir dille anlatmak gerekirse bu adımdaki amaç; sınıflandırmak istenilen x^* verisini merkez olarak kabul edilip, belli bir yarıçap içinde hangi sınıftan daha çok sayıda veri varsa, çok olan verilerin sınıfına atanması işlemidir. Buradaki yarıçap ise belirlenen K değerine göre değişecektir. Aşağıda da burada basit bir dille anlattığımız sınıflandırma işleminin matematiksel ifadesi yer almaktadır.

- $c_i^{'}$: x^* verisinin i. komşuluğundaki verinin sınıfı

- $c_j^{'}$: $c_i^{'} \neq c_j^{'}$ olacak şekilde en yakın K komşu verilerinin bilinen sınıfları
- $g(c_i^{'})$: Her bir $c_j^{'}$ sınıfının sayısını veren bir fonksiyon

olmak üzere herhangi bir x^* verisinin c^* sınıfı şu şekilde belirlenir:

$$I(c_j^{'}, c_i^{'}) = \begin{cases} 1 & c_j^{'} = c_i^{'} \\ 0 & c_j^{'} \neq c_i^{'} \end{cases} \tag{5.5}$$

birim fonksiyonu yardımıyla her bir $c_i^{'}$ sınıfının sayısı

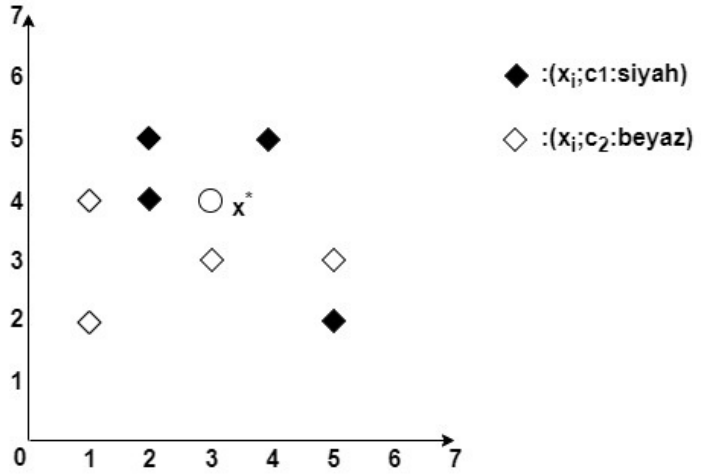
$$g(c_i^{'}) = \sum_{c_j^{'} \neq c_i^{'}} \sum_{i=1}^{K} I(c_j^{'}, c_i^{'}) \tag{5.6}$$

denklemi ile bulunur. Bulunan her bir $c_i^{'}$ sınıflarının K tane komşu içinde sayıca en fazla olanı

$$c^* = argmax g(c_i^{'}) \tag{5.7}$$

denklemi yardımıyla x^* verisinin c^* sınıfı olarak belirlenir.

Algoritmayı daha iyi kavrayabilmek adına aşağıda verilen veri seti üzerinde uygulamasını yapalım.



Şekil 5.4 : İki boyutlu düzlemde etiketli veriler ve etiketsiz verinin görünümü.

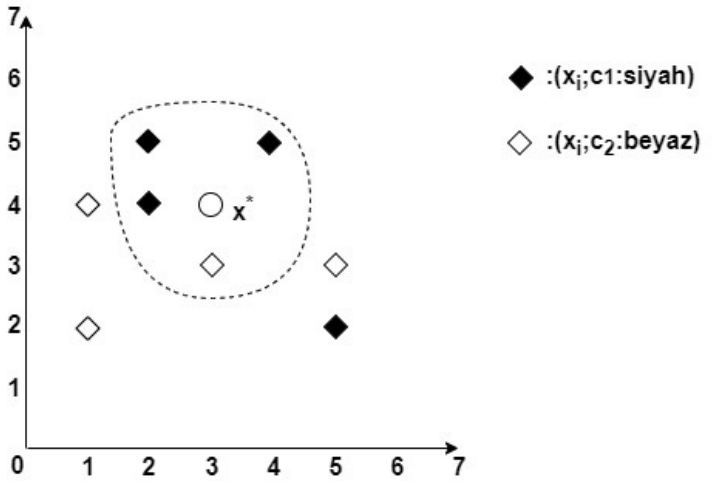
1.Adım:

Sınıflandırmak istediğimiz $x^* = (3,4)$ verisinin diğer veriler ile olan uzaklığını hesaplayalım. $K = 4$ seçilmiş olsun.

Çizelge 5.4 : x^*' a olan uzaklıklar.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x*	2	2	1	$\sqrt{2}$	1	$\sqrt{2}$	$2\sqrt{2}$	$\sqrt{5}$

$K = 4$ seçildiğinden yukarıdaki uzaklıklardan sınıflandırmak istediğimiz $(x^*;c^*)$ verisine en yakın 4 tane veri x_3,x_4,x_5,x_6 olacaktır. 4 tane en yakın veriyi kapsayacak şekilde $(x^*;c^*)$ merkezli bir daire çizelim. Bu daireyi çizmekteki amacımız; bu daire içinde kalan verilerin sınıflarını saymak ve sayıca çok olan sınıfı x^* verisinin sınıflandırmak istediğimiz sınıfı c^*' ın sınıfı olarak belirlemek.



Şekil 5.5 : Sınıflandırılmak istenen x^* verisine en yakın veriler.

2.Adım:

Aşağıdaki Şekil (5.5)' de çizilen daire içinde x^* verisi merkez kabul edilip bir daire çizilmiştir.

Bu daire içinde basit bir sayma işlemi yaparsak, verilerden üç tanesinin sınıfı *siyah* geriye kalan bir tanesinin sınıfının ise *beyaz* olduğunu görürüz. Bu bölümün başında da belirttiğimiz gibi K-NN algorimasında sınıflandırma işlemi bir daire içindeki sınıfları sayma işleminden ibarettir. Bu bilgiler ışığında örneğimize dönecek olursak; x^* verisinin c^* sınıfı *siyah* olarak belirlenecektir.

5.4 Naive Bayes Sınıflandırıcı *(Naive Bayes Classifier)*

Naive Bayes Sınıflandırıcı'dan bahsetmeden önce bu sınıflandırıcının temeli olan Bayes Teoremi'nden bahsetmek gerekir.

5.4.1 Bayes Teoremi

X ve Y birbirinden bağımsız rasgele iki olay olsun. Thomas Bayes tarafından ortaya atılan Bayes Teoremi, bir X olayının gerçekleşmesi durumunda Y olayının gerçekleşme olasılığını hesaplar.

1. $P(X)$: Bir X olayının olasılığını
2. $P(Y)$: Bir Y olayının olasılığını
3. $P(Y|X)$: Bir X olayı gerçekleştiğinde Y olayının olma olasılığı

olmak üzere Bayes Teoremi,

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{5.8}$$

şeklinde ifade edilir.

5.4.2 Naive Bayes sınıflandırıcı

Naive Bayes Sınıflandırıcı, bir $X = \{x_1, x_2, ..., x_n\}$ verisine karşılık bir C sınıf özniteliğinde belirli bir sınıfın gerçekleşme durumunu tahmin eder. Bu tahminleme işlemi yapılırken Bayes Teoremi'nden yararlanılır. Büyük boyutlu verilerde iyi performans vermez.

Naive Bayes Sınıflandırıcı, Bayes Teoremi'nden farklı olarak sınıfı bilinmeyen bir X verisinin sınıfını tahmin etmeye odaklanır. Bayes Teoremi'nde ise olasılık hesabı yapılır.

n tane niteliğe (değere) sahip sınıfı bilinmeyen bir $X = \{x_1, x_2, ..., x_n\}$ verisi olsun. $C = \{c_1, c_2, ..., c_k\}$ olası k tane sınıfı temsil eden bir sınıf özniteliği ise, sınıfı belirlenecek olan X verisi için,

$$P(c_k|X) = \frac{P(X|c_k)P(c_k)}{P(X)} \tag{5.9}$$

olasılıkları hesaplanır.

Denklem (5.9) yardımıya bulunan olasılıklardan aşağıdaki denklem yardımıyla maximum olanı seçilerek sınıfı bilinmeyen X verisinin sınıfı belirlenmiş olur.

$$argmax_{c_k}\{P(c_k|X)P(X)\} \tag{5.10}$$

Bu ifade en büyük sonsal sınıflandırma (*Maximum A Posteriori Classification*) olarak da bilinir [13]. Fakat $X = \{x_1, x_2, ..., x_n\}$ verisinin birden çok değere (n>1) sahip olduğu durumda, Bayes Formülü'nün kullanımı değişir ve X verisinin sınıf tahmnini yapmak için X'in tüm değerlerinin koşullu olasılıkları hesaplanmalıdır.

$$P(x_1, x_2, ..., x_n|c_k) = \prod_{i=1}^{n} P(x_i|c_k) \tag{5.11}$$

O halde birden çok niteliğe sahip X verisini sınıflandırmak için

$$C = argmax_{c_k \in C}(P(C = c_k) \prod_{i=1}^{n} P(X = x_i|C = c_k)) \tag{5.12}$$

denklemi kullanılır.

Naive Bayes Yöntemi kullanarak yapılan sınıflandırmayı daha iyi ifade edebilmek adına Quinlan tarafından ortaya atılan ve aynı zamanda klasik bir örnek olan *Weather* örnek veri kümesi ile algoritmayı uygulayalım.

Çizelge 5.5 : Weather veri kümesi.

HAVA	SICAKLIK	NEM	RÜZGAR	YELKEN
Bulutlu	Sıcak	Yüksek	Hafif	Hayır
Güneşli	Sıcak	Yüksek	Hafif	Hayır
Güneşli	Sıcak	Yüksek	Kuvvetli	Hayır
Yağmurlu	Ilık	Yüksek	Hafif	Evet
Yağmurlu	Soğuk	Normal	Hafif	Evet
Yağmurlu	Soğuk	Normal	Kuvvetli	Hayır
Bulutlu	Soğuk	Normal	Kuvvetli	Evet
Güneşli	Soğuk	Normal	Hafif	Evet
Yağmurlu	Ilık	Normal	Hafif	Evet
Güneşli	Ilık	Normal	Hafif	Evet
Yağmurlu	Ilık	Yüksek	Kuvvetli	Hayır
Güneşli	Ilık	Yüksek	Hafif	Hayır
Bulutlu	Sıcak	Normal	Hafif	Evet
Bulutlu	Ilık	Yüksek	Kuvvetli	Evet

Yukarıdaki *Weather* örnek veri kümesini eğitim kümesi olarak ele alıp Bayes Sınıflandırıcı kullanarak $x_1 : HAVA =$ Bulutlu, $x_2 : SICAKLIK =$ Sıcak, $x_3 : NEM =$ Normal, $x_4 : RUZGAR =$ Kuvvetli test verisi ile $YELKEN$ sınıf özniteliğini tahmin etmeye çalışalım. Eğitim kümesinde verilen verilerin olasılık tablosunu yapalım:

Öncelikle her bir hipotez için yani $C_1 : YELKEN = EVET$ ve $C_2 : YELKEN = HAYIR$ için Bayes olasılıkları hesaplanmalıdır. Öncelikle Denkem (5.9)'da

Çizelge 5.6 : Weather veri kümesi olasılık tablosu.

		YELKEN			
Öznitelikler	**Nitelikler**	**EVET**		**HAYIR**	
		Sayısı	**Olasılık**	**Sayısı**	**Olasılık**
HAVA	Bulutlu	3	3/8	1	1/6
	Güneşli	2	2/8	3	3/6
	Yağmurlu	3	3/8	2	2/6
SICAKLIK	Ilık	4	4/8	2	2/6
	Sıcak	1	1/8	3	3/6
	Soğuk	3	3/8	1	1/6
NEM	Yüksek	2	2/8	5	5/6
	Normal	6	6/8	1	1/6
RÜZGAR	Kuvvetli	2	2/8	3	3/6
	Hafif	6	6/8	3	3/6

verilen $P(X|c_1)P(c_1)$ ve $P(X|c_2)P(c_2)$ olasılıkları hesaplanarak bu iki olasılıktan hangisinin büyük olduğu denklem (5.10) yardımıyla bulunur. Burada bizim X verimiz birden çok niteliğe sahip olduğundan X verisinin denklem (5.11) yardımıyla koşulluklu olaslılıkları hesaplanmalıdır. Ardından aynı denklem (5.10)'daki gibi denklem (5.12) kullanılarak en büyük olasılığa sahip olan sınıf bulunabilir.

$c_1 = Evet$ için $P(X|c_1)P(c_1)$ olasılığının hesaplanması:

$P(X|c_1)P(c_1)$ olasılığını hesaplayabilmek için öncelikle $P(X|YELKEN = Evet)$ koşullu olasılığı hesaplanmalıdır. X özniteliğinin birden çok niteliği olduğundan $(X = x_1, x_2, x_3, x_4)$ bu koşullu olasılığı denklem (5.11) yardımıyla hesaplayalım:

Olasılık tablosu yardımıyla;

- $P(x_1|c_1) = P(HAVA = Bulutlu|YELKEN = Evet) = \frac{3}{8}$
- $P(x_2|c_1) = P(SICAKLIK = Sıcak|YELKEN = Evet) = \frac{1}{8}$
- $P(x_3|c_1) = P(NEM = Normal|YELKEN = Evet) = \frac{3}{4}$
- $P(x_4|c_1) = P(RUZGAR = Kuvvetli|YELKEN = Evet) = \frac{1}{4}$

olduğu görülür. Denklem (5.11) yardımıyla $P(X|c_k) = P(X|YELKEN = Evet)$ koşullu olasılığı;

$$P(X|c_k) = P(X|YELKEN = Evet) = \frac{3}{8}.\frac{1}{8}.\frac{3}{4}.\frac{1}{4} = \frac{9}{1024}$$

bulunur. Fakat bizim $P(c_1)$ olasılık değerine de ihtiyacımız var.

$$P(c_1) = P(YELKEN = Evet) = \frac{4}{7}$$

olduğu kolayca görülür. Böylece bulduğumuz olasıklar denklem (5.12)' de yerine konulursa;

$$\begin{aligned} P(X = x_1,...,x_4|c_1)P(c_1) =& P(X|YELKEN = Evet)P(YELKEN = Evet) \\ =& \frac{9}{1024}.\frac{4}{7} \\ =& \frac{9}{1792} \end{aligned}$$

ilk hipotemizin olan $c_1 = Evet$ için olasılık bulunmuş oldu.

O halde $c_2 =$ Hayır için $P(X|c_2)P(c_2)$ olasılığını aynı adımları izleyerek hesaplayalım.

Olasılık tablosu yardımıyla

- $P(x_1|c_2) = P(HAVA = Bulutlu|YELKEN = \text{Hayır}) = \frac{1}{6}$
- $P(x_2|c_2) = P(SICAKLIK = \text{Sıcak}|YELKEN = \text{Hayır}) = \frac{1}{2}$
- $P(x_3|c_2) = P(NEM = Normal|YELKEN = \text{Hayır}) = \frac{1}{6}$
- $P(x_4|c_2) = P(RUZGAR = Kuvvetli|YELKEN = \text{Hayır}) = \frac{1}{2}$

olduğu görülür. Denklem (7.11) yardımıyla $P(X|c_k) = P(X|YELKEN = \text{Hayır})$ koşullu olasılığı;

$$\begin{aligned} P(X|c_k) &= P(X|YELKEN = \text{Hayır}) \\ &= \frac{1}{6}.\frac{1}{2}.\frac{1}{6}.\frac{1}{2} = \frac{1}{144} \end{aligned}$$

bulunur. Fakat bizim $P(c_2)$ olasılık değerine de ihtiyacımız var.

$$P(c_2) = P(YELKEN = \text{Hayır}) = \frac{3}{7}$$

olduğu kolayca görülür. Böylece bulduğumuz olasıklar denklem (5.12)' de yerine konulursa;

$$P(X = x_1, ..., x_4|c_1)P(c_2) = P(X|YELKEN = \text{Hayır})P(YELKEN = \text{Hayır})$$
$$= \frac{1}{144}.\frac{3}{7} = \frac{3}{1008}$$

Böylece ikinci hipotemizin olan $c_2 =$ Hayır için olasılığımızı da bulmuş olduk.

Her iki hipotez için de olasılıklar bulunduğuna göre denklem (5.12) yardımıyla bu olasılıklardan en büyük olanı seçilir ve test verimiz olasılığı büyük olan sınıfa atanır.

$k = 1,2$ ve test verimizin tespit edilmeye çalışılan sınıfı C olmak üzere;

$$C = argmax_{c_k} P(X|c_k)P(c_k) = \{0.005; 0.002\}$$

olarak bulunur. Burada görülüyor ki; $c_1 = Evet$ hipotezi için olasılık daha fazladır. Dolayısıyla test verimizin $YELKEN$ sınıfı *Evet* olarak belirlenir.

Eğer Naive Bayes ile sınıflandırma işleminde olasılık tablosundaki herhangi bir olasılık sıfır çıkıyorsa, sıfır yerine sıfıra çok yakın sıfır olmayan bir sayı atanarak işleme devam edilir. Böylece koşullu olasılığın sıfır olma durumu ortadan kalkmış olur.

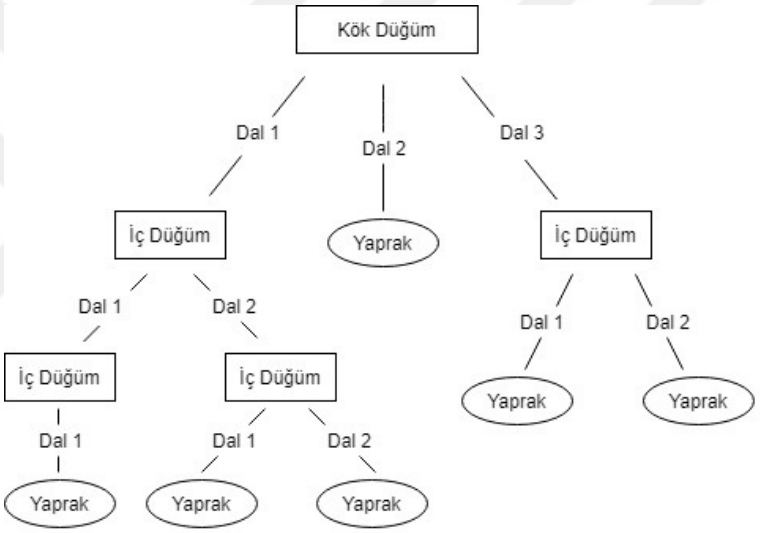
5.5 Karar Ağaçları *(Decision Trees)*

Karar ağaçları bir sınıflandırma algoritması olup, gözetimli öğrenme algoritmaları sınıfına dahildir.

Karar ağaçları gözetimli öğrenme algoritmalarına dahil oluduğundan iki aşamalı olarak gerçekleştirilir. İlk aşama öğrenme olarak adlandırılan model oluşturma aşamasıdır. İkinci aşama ise test kümesindeki veriler model üzerinde uygulanır ve test kümesindeki her bir verinin sınıfının belirlenmesi amaçlanır. [12]

Karar ağaçlarında bazı özel terimler kullanılmaktadır. Bunlar; düğüm (*node*), dal (*branch*) ve yaprak (*leaf*)'tır. Karar ağaçlarının oluşturulmasında genellikle yukarıdan aşağıya (*top-down*) bir yapı izlendiğinden, ağacın üst kısmında yer alan düğüme kök düğüm (*root node*), ağacın yaprakları ile kök düğüm arasında yer alan düğümlere ise iç düğüm (*internal node*) adı verilir.

Her düğüm veri kümesindeki farklı bir özniteliği temsil eder. Veri kümesindeki her bir sınıf ise yapraklar tarafından temsil edilir. Dallar ise bir öznitelikteki değişkenleri temsil eder. Yani bir özniteliğin boyutu ne kadar fazlaysa dallanma o kadar fazla olacaktır. Dallanma işlemi yaprak düğümüne ulaşıncaya kadar devam eder.



Şekil 5.6 : Örnek bir karar ağacı yapısı.

Karar ağaçlarında amaç, bir veri kümesini bazı kriterler yardımıyla olabilecek en iyi şekilde sınıflandırmak ve olabildiğince küçük kümelere ayırmaktır [20].

5.5.1 Karar ağaçlarında dallanma

Karar ağaçlarını yapıları gereği basit bir veri akış şemasına benzetebiliriz. Karar ağaçları bu benzerliklerinden dolayı diğer sınıflandırma algoritmalarına göre daha anlaşılabilir bir yapıdadır. Fakat daha anlaşılabilir olması bazı sorunları beraberinde getirmiyor değil. Karar ağaçları oluşturulurken karşılaşılan en büyük sorun, ağaçta dallanmanın veri kümesindeki hangi öznitelik kullanılarak başlatılacağıdır. Bir başka deyişle kök düğümün seçimidir. Karar ağaçlarında bu sorun entropi kavramı ile aşılmaktadır.

5.5.2 Entropi

Entropi genel olarak bir sistemdeki düzensizliğin, safsızlığın ölçüsü olarak tanımlanır [21].

$S = \{s_1, s_2, ..., s_n\}$ bir veri kümesi olsun. Her $s_i \in S$'nin $j \leq k$ ve $s_i \in C_j$ olmak üzere $C = \{c_1, c_2, ..., c_k\}$ olmak üzere k tane sınıfa ayrıldığını varsayalım. O halde Her $s_i \in S$ olmak üzere s_i değerine ait olasılık

$$p_i = (\frac{|C_i|}{|C|}) \tag{5.13}$$

olarak ifade edilir. Buna bağlı olarak olasılık dağılımı

$$P = (p_1, p_2, ..., p_k) \tag{5.14}$$

olarak ifade edilebilir. Bu bilgiler ışığında S kümesinin entropisi

$$H(S) = -\sum_{i=1}^{n} p_i.log_2 p_i \tag{5.15}$$

denklemi yardımıyla hesaplanır. Eğer $C = \{c_1, c_2, ..., c_k\}$ sınıf kümesi için

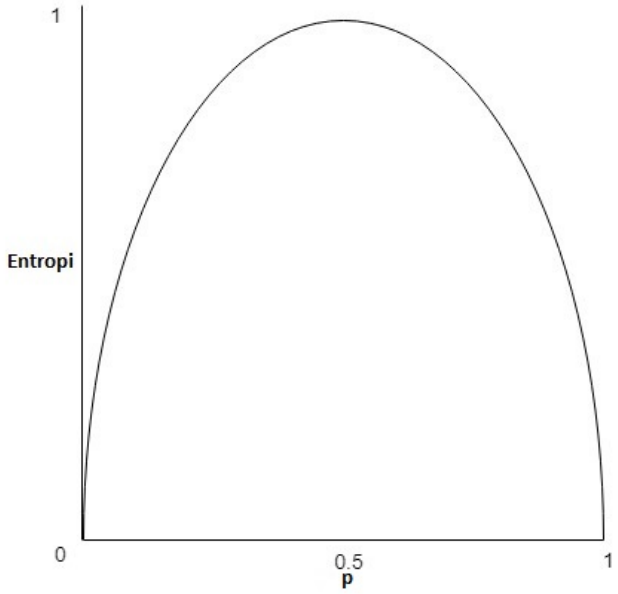
- $k \leq 2 \rightarrow$ Entropi değeri [0,1] aralığında değer alacaktır.
- $k > 3 \rightarrow$ Entropi değeri [0,1] aralığında değer almayabilir.

Entropi değerinin küçük olması düzensizliğin az olduğu anlamına gelir. Karar ağaçlarında da entropi değeri en küçük olan öznitelik seçilerek dallanma yapılır. Aşağıda iki olası sınıflı bir veri kümesi için entropi değerinin olasılığa bağlı olarak nasıl değiştiği görülmektedir.

5.5.3 ID3 (*Iterative Dichotomiser 3*) algoritması

ID3 Algoritması, 1986 yılında Quinlan tarafından karar ağaçlarında sınıflandırma işlemlerini yerine getirmek üzere geliştirilmiş bir algoritmadır [22]. Sadece kategorik verilerle çalışır.

Bir veri kümesi, bir özniteliğe göre bölündüğünde bir başka deyişle dallandığında elde edilen her bir alt veri kümesinin entropisi (*düzensizliği*) minimum ve entropi yardımıyla elde edilen bilgi kazancı fonksiyonu (*Gain function*) maksimum ise bölünme için olabilecek en iyi öznitelik seçimi yapılmış demektir. [22]. Bu bilgi



Şekil 5.7 : İki olası sınıflı bir veri kümesi için entropi.

ışığında ID3 Algoritması ile sınıflandırma yapılırken her bir düğüme karşılık gelen özniteliğin bilgi kazancı fonksiyonunun maksimum olması hedeflenir.

5.5.3.1 Dallanma için uygun öznitelik seçimi

Her $x_i \in \mathbb{R}^p$, (i=1,2,...,n) olsun. $C = \{c_1, c_2, ..., c_k\}$ olacak şekilde her $c_k \in C$, $x_i \in D$ verisinin bir sınıfını temsil etsin. O halde eğitim kümesi $D_{egitim} = \{x_i; c_k\}$ olarak yazılabilir.

- Her $|C_i|$: i.sınıfın kaç adet olduğunu
- $|C|$: sınıf kümesinde toplam kaç tane sınıf olduğunu

göstermek üzere sınıf değerlerini barındıran C sınıf kümesi için olasılık dağılımı

$$p_i = (\frac{|c_1|}{|C|}, \frac{|c_2|}{|C|}, ..., \frac{|c_k|}{|C|}) \tag{5.16}$$

olarak ifade edilir. O halde C kümesi için entropi

$$H(C) = -\sum_{i=1}^{n} p_i.log_2 p_i \tag{5.17}$$

olarak hesaplanır. C sınıf özniteliğine bağlı olarak her bir X özniteliğinin entropisi ise

$$H(X,C) = \sum_{i=1}^{n} \frac{|c_i|}{|C|}.H(c_i) \tag{5.18}$$

şeklindedir. Denklem (5.18) yardımıyla dallanma için bir X özniteliği seçildiğinde ne kadar kazanç sağlandığı bilgi kazancı fonksiyonu

$$Gain(X,C) = H(C) - H(X,C) \tag{5.19}$$

yardımıyla hesaplanır. Bilgi kazancını maximum yapan X özniteliği seçilerek ağaç dallandırılır.

Algoritmayı daha iyi ifade edebilmek adına yine *Weather* örnek veri kümesi üzerinde algoritmayı uygulayalım.

Çizelge 5.7 : Weather veri kümesi.

HAVA	SICAKLIK	NEM	RÜZGAR	YELKEN
Bulutlu	Sıcak	Yüksek	Hafif	Evet
Güneşli	Sıcak	Yüksek	Hafif	Hayır
Güneşli	Sıcak	Yüksek	Kuvvetli	Hayır
Yağmurlu	Ilık	Yüksek	Hafif	Evet
Yağmurlu	Soğuk	Normal	Hafif	Evet
Yağmurlu	Soğuk	Normal	Kuvvetli	Hayır
Bulutlu	Soğuk	Normal	Kuvvetli	Evet
Güneşli	Soğuk	Normal	Hafif	Evet
Yağmurlu	Ilık	Normal	Hafif	Evet
Güneşli	Ilık	Normal	Hafif	Evet
Yağmurlu	Ilık	Yüksek	Kuvvetli	Hayır
Güneşli	Ilık	Yüksek	Hafif	Hayır
Bulutlu	Sıcak	Normal	Hafif	Evet
Bulutlu	Ilık	Yüksek	Kuvvetli	Evet

Yelken veri kümesi bizim sınıf değişkenlerimizi içermektedir. c_1 : evet ve c_2 : hayır olmak üzere iki farklı sınıf değişkenimiz vardır. Sınıf özniteliğimize göre entropi hesabı denklem (5.15) yardımıyla

$$H(C = YELKEN) = -(\frac{5}{14} log_2 \frac{5}{14} + \frac{9}{14} log_2 \frac{9}{14}) = 0.940 \tag{5.20}$$

olarak bulunur.

Entropiyi bulduk. Şimdi karar ağacını oluşturmak için hangi özniteliğin kök düğüm olarak seçilmesi gerektiğini bulmamız gerekiyor. Bunun için bulduğumuz $H(C = YELKEN) = 0.940$ entropi değerini kullanarak veri setindeki tüm öznitelikler için bilgi kazancını hesaplamalıyız. Amacımız bize en çok bilgi kazancını sağlayacak olan özniteliği seçmektir [23].

SICAKLIK özniteliğinin bilgi kazancına bakalım. Bu özniteliğin bilgi kazancını hesaplayabilmek için bu özniteliğin farklı değerlerinin sayısına ihtiyacımız vardır.

$$|\text{SICAKLIK}_{\text{soğuk}}| = 4, |SICAKLIK_{\text{ılık}}| = 6, |SICAKLIK_{\text{sıcak}}| = 4$$

şeklindedir. Eğer SICAKLIK özniteliğine göre bir dallanma yapılacaksa bu seçim sonucunda bilgi kazancının ne olacağını hesaplamak gerekir. Denklem (5.19) bilgi kazancını hesaplamak için kullanılmaktadır. Fakat daha önce SICAKLIK özniteliğindeki her bir değer için entropiyi hesaplamalıyız. Daha sonra da

$$H(X,C) = \sum_{i=1}^{n} \frac{|c_i|}{|C|}.H(c_i) \tag{5.21}$$

denklemini hesaplamak gerekiyor. Her bir $|c_i|$ değeri, SICAKLIK özniteliğindeki her farklı değerin kaç tane olduğunu ifade etmektedir. O halde

$$H(\text{S=SICAKLIK}, \text{Y=YELKEN}) = \frac{1}{14}.H(\text{S}_{\text{soğuk}}) + \frac{6}{14}.H(S_{\text{ılık}}) + \frac{4}{14}.H(S_{\text{sıcak}}) \tag{5.22}$$

Denklemde görülen SICAKLIK özniteliği altında yer alan değişkenler için $H(S_{\text{soğuk}})$, $H(S_{\text{ılık}})$, $H(S_{\text{sıcak})}$ entropileri hesaplanır;

$$\begin{aligned} H(S_{\text{soğuk}}) &= -(\tfrac{1}{4}log_2\tfrac{1}{4} + \tfrac{3}{4}log_2\tfrac{3}{4}) = 0.811 \\ H(S_{\text{ılık}}) &= -(\tfrac{2}{46}log_2\tfrac{2}{6} + \tfrac{4}{6}log_2\tfrac{4}{6}) = 0.918 \\ H(S_{\text{sıcak}}) &= -(\tfrac{2}{4}log_2\tfrac{2}{4} + \tfrac{2}{4}log_2\tfrac{2}{4}) = 1 \end{aligned}$$

O halde $H(S,Y)$ için entropi;

$$H(S,Y) = \frac{1}{14}.(0.811) + \frac{6}{14}.(0.918) + \frac{4}{14}.1 = 0.911 \tag{5.23}$$

olarak hesaplanır.

Artık elimizde SICAKLIK özniteliğinin her bir değeri için entropiler olduğuna göre, sıra SICAKLIK özniteliğinin seçilmesi durumunda ne kadar bilgi kazancı getireceğini hesaplamaya geldi. Bunun için

$$Gain(X,C) = H(C) - H(X,C) \tag{5.24}$$

denklemini kullanalım. O halde dallanma için SICAKLIK özniteliğinin seçilmesi durumunda elde edeceğimiz bilgi kazancı:

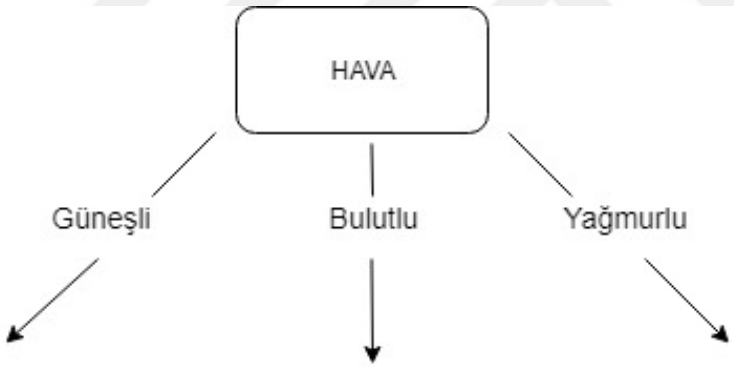
$$Gain(S,Y) = 0.940 - 0.911 = 0.029 \tag{5.25}$$

olarak hesaplanır.

Benzer şekilde her bir öznitelik için bilgi kazancı hesaplanırsa;

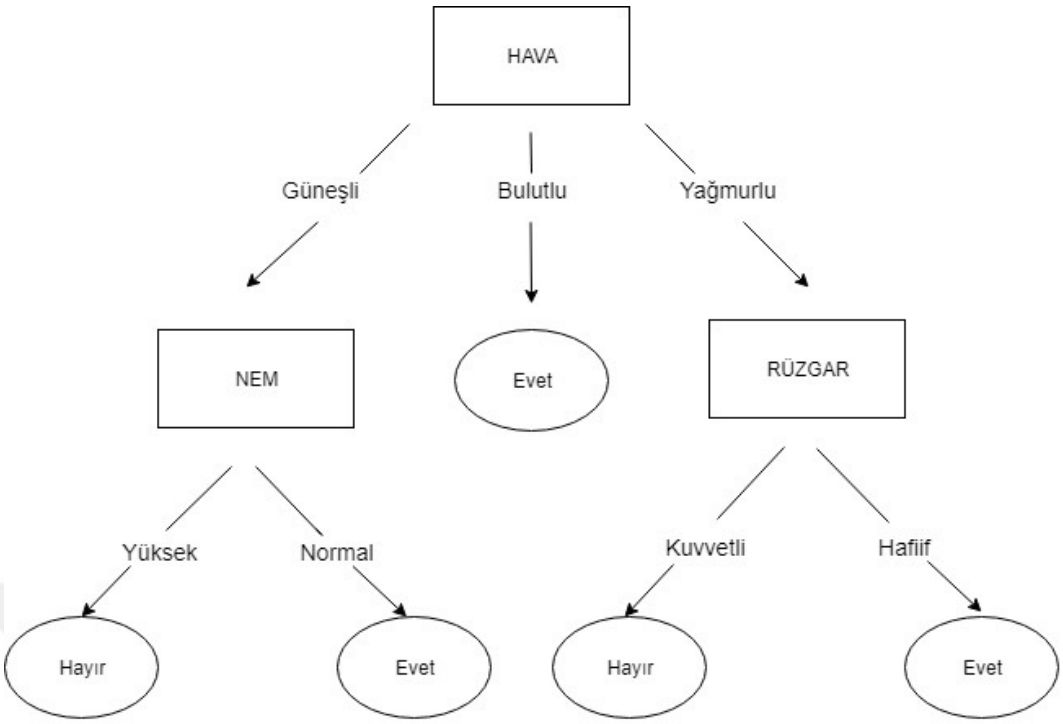
$$Gain(HAVA,YELKEN) = 0.247$$
$$Gain(NEM,YELKEN) = 0.151$$
$$Gain(HAVA,YELKEN) = 0.048$$

olarak bulunur. Görüldüğü gibi bilgi kazancı olarak HAVA özniteliği ile yapılan dallanma ile gerçekleşecektir. Dolayısıyla karar ağacı için ilk düğümümüz yani kök düğümümüz HAVA özniteliği olarak belirlenir. Böylece karar ağacında 1. dallanma aşağıdaki gibi olacaktır.



Şekil 5.8 : Karar ağacında ilk dallanma.

Her bir dal yaprağa ulaşana kadar yukarıdaki adımlar izlenerek dallandırma işlemine devam edilir. Her bir dal yaprağa ulaştığında karar ağacımız tamamlanmış demektir. Aşağıda *weather* veri kümesi için tamamlanmış karar ağacı görülmektedir.



Şekil 5.9 : *weather* veri kümesi için karar ağacı.

5.5.4 C4.5 algoritması

ID3 Algoritması'nda dallanma kriteri olarak kullanılan bilgi kazancı, çok sayıda farklı değere sahip özniteliklere karşı düşük performanslı çalışabilir. ID3 Algoritması'ndan farklı olarak kategorik değişkenlerin yanı sıra sürekli değişkenler ile de çalışan C4.5 Algoritması, Quinlan tarafından ID3 Algoritması'nın bu tip eksik yönlerini gidermek amacıyla geliştirilmiştir.

C4.5 Algoritması'da ID3 Algoritması gibi entropi tabanlı bir algoritmadır. Yani dallanma kriteri için entropiye değerlerine ihtiyaç duymaktadır.Fakat ID3 Algoritması'ndan farklı olarak dallanma kriteri olarak kazanç oranı(*Gain Ratio*) kullanılmaktadır.

Öncelikle bilgi kazancına bölme bilgisi(*split information*) olarak adlandırılan bir normalizasyon uygulanmaktadır. Bu sayede aykırı verilerin sınıflandırmaya etkisi azaltılmaktadır. [11]

$$SplitInfo(X,C) = -\sum_{i=1}^{n} \frac{|c_i|}{|C|}.log_2(\frac{|c_i|}{|C|}) \tag{5.26}$$

olarak tanımlanan bölme bilgisi; eğitim kümesindeki X özniteliğinin aldığı n tane farklı değere bağlı olarak n parçaya ayrıldığında oluşturulan potansiyel bilgiyi

temsil etmektedir. Buna bağlı olarak kazanç oranı

$$GainRatio(X,C) = \frac{Gain(X,C)}{SplitInfo(X,C)} \tag{5.27}$$

olarak ifade edilir. Her X özniteliği için kazanç oranı hesaplandıktan sonra en yüksek kazanç oranını sağlayan X özniteliği dallanmanın yapılacağı öznitelik olarak seçilir.

5.6 Logistik Regresyon *(Logistic Regression)*

5.6.1 Kategorik veriler için olasılık dağılımları

Bu bölümde, kategorik veriler için önemli dağılımlar olan *Binom Dağılım* ve *Multinomiyal (Çoklu) Dağılım* incelenecektir.

5.6.1.1 Binom dağılım

Bağımlı değişkenin iki olası kategorik veri tipli sonuca sahip olması durumda binom dağılım kullanılmaktadır.

Birbirinden bağımsız ve eşit olasılıklı n tane gözlem $y_1, y_2, ..., y_n$ şeklinde gösterilsin. Bağımlı değişkenimizin alabileceği olası sonuçlar "başarılı" ve "başarısız" olarak ifade edilsin. Bu sonuçları sırasıyla "1" ve "0" olarak ifade edelim. Bu bilgiler ışığında:

- π : Başarı olasılığını
- $P(Y_i = 1)$=π : Başarılı olma olasılığını
- $P(Y_i = 0)$=1-π : Başarısız olma olasılığını

göstermektedir.

Y kategorik veri tipli bağımlı değişken için olası y tane sonucu gösteren olasılık yoğunluk fonksiyonu;

$$P(Y) = \binom{n}{y} \pi^y . (1-\pi)^{n-y}, y = 0, 1, 2, ..., n \tag{5.28}$$

şeklinde ifade edilir [10].

5.6.1.2 Multinomiyal (Çoklu) dağılım

Bazı durumlarda bağımlı değişken ikiden fazla olası kategorik veri tipli sonuca sahip olabilir. Bu durumda binom dağılım yerine multinomiyal dağılım kullanmak gerekecektir.

k tane farklı kategorik sonuca sahip, eşit olasılıklı ve birbirinden bağımsız n tane deneme varsayalım. Öyle ki herhangi bir $i.$ deneme k tane farklı kategori arasından herhangi bir $c.$ kategoriye eşitse, $y_{ic} = 1$ aksi taktirde $y_{ic} = 0$ olarak ifade edilir. Bu bilgiler doğrultusunda,

$$y_i = (y_{i1}, y_{i2}, ..., y_{ik}) \tag{5.29}$$

multinomiyal(çoklu) bir deneme olarak ifade edilir [10]. Bu durumda $\sum_c y_{ic} = 1$ olur.

5.6.2 Regresyon

Genel olarak regresyon analizi, bağımlı değişkenler ile bağımsız değişkenler arasında bir ilişki kurmayı amaçlayan bir yöntemdir. Başka bir amacı ise sınıflandırma olarak düşünülebilir.

$$Y = \beta_0 + \beta_1 x_1 \tag{5.30}$$

$$Y = \beta_0 + \beta_1 x_1^2, \qquad Y = \frac{1}{\beta_0 + \beta_1 x_1} \tag{5.31}$$

Regresyon analizinde bağımlı değişken ile bağımsız değişkenler arasındaki ilişki her zaman Denklem (5.30)' teki gibi *doğrusal (lineer)* olmayabilir. Bazı durumlarda da Denklem (5.31)' teki gibi doğrusal olmayan durumlarla da karşılaşılabilir. O halde regresyon analizi, doğrusal regresyon ve doğrusal olmayan regresyon olarak iki ayrı alt başlık altında incelenebilir. Biz bu çalışmada bir doğrusal olmayan regresyon modeli kullanarak sınıflandırma yapan logistik regresyonu inceleyecegiz.

Regresyon analizinde bağımlı değişken her zaman sürekli *nicel, nümerik* yapıda olmayabilir. Bu tip bir durumda lineer regresyon yetersiz kalacaktır. Bu sorunu aşmak adına, yani bağımlı değişkenin kesikli *(nitel, kategorik)* olması durumda, logistik regresyon yöntemi geliştirilmiştir [24].

5.6.3 Logistik regresyon

'Siyah' ve 'Beyaz' gibi iki olası kategorik bağımlı değişken olsun. O halde logistik regresyonda sınıflandırma işlemi, bir x verisinin Siyah ya da Beyaz sınıfına ait olma olasılığı tahmin etmek üzerine kuruludur. Logistik regresyon bağımlı değişkenler ile bağımsız değişkenler arasında *logit* bir ilişki olduğunu varsayar ve bazı logaritmik dönüşümler yardımıyla doğrusal olmayan bağımlı değişken ve bağımsız değişkenler arasındaki ilişkiyi doğrusal bir yapıya getirir.

Logistik regresyonda, bağımlı değişken iki olası kategoriye sahipse *İkili Logistik Regresyon (Binary Logistic Regression)*, bağımlı değişken ikiden fazla olası kategoriye sahipse *Çoklu Logistik Regresyon (Multinomial Logistic Regression)* olarak adlandırılır.

Regresyon analizinde Y bağımlı değişken ve x bağımsız değişken olmak üzere, x verildiğinde Y' nin koşullu ortalaması *(beklenen değeri)* $E(Y|x)$ olarak ifade edilir. Daha önce de belirtildiği gibi doğrusal regresyon analizinde bu koşullu ortalama $E(Y|x) = \beta_0 + \beta_1 x_1$ şeklinde x' e göre doğrusal bir şekilde ifade edilir. Buna karşın, logistik regresyon analizinde $E(Y|x)$ koşullu ortalaması $0 \leq E(Y|x) \leq 1$ arasında bir olasılık değeri alır [25].

$p(x)$ bir x olayının gerçekleşme olasılığını ve $1 - p(x)$' de aynı olayın gerçekleşmeme olasılığını göstermek üzere Y' nin koşullu ortalaması

$$p(x) = E(Y|x) \tag{5.32}$$

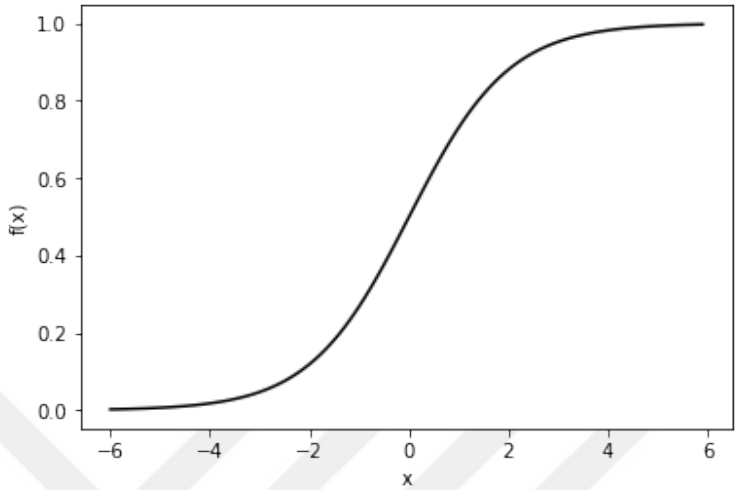
şeklinde tanımlanabilir [26].

Bu bilgiler ışığında $p(x)$ ile x bağımsız değişkeni arasında ilişkiyi kuran logistik regresyon modeli

$$p(x) = E(Y|x) = \frac{e^{\beta_0+\beta_1 x_1}}{1+e^{\beta_0+\beta_1 x_1}} = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1)}} \tag{5.33}$$

denklemi ile ifade edilir ve bu denklem logistik fonksiyon olarak adlandırılır [24].

$\beta_0 + \beta_1 x_1 = z$ dönüşümü yapılırsa logistik fonksiyon $E(Y|x) = \frac{1}{1+e^{-z}}$ şeklinde de ifade edilebilir.



Şekil 5.10 : Sigmoid fonksiyonu grafiği.

Bağımlı değişkenin olası yine *Siyah* ve *Beyaz* gibi iki olası kategoriye sahip olduğu düşünülürse, söz konusu x öznitelikleri kullanılarak oluşturulan logistik regresyon modelinden çıkan olasılık değeri, yani $E(Y|x) > 0.5$ ise *Siyah*, $E(Y|x) < 0.5$ ise *Beyaz* olarak etiketlenir.

Logistik regresyonda sınıflandırma yapmak için seçilen fonksiyon 0 ve 1' e çok hızlı yakınsayabiliyor olmalıdır. Bu tez kapsamında sigmoid fonksiyonu $f(x) = \frac{1}{1+e^{-x}}$ kullanıldı. Fakat sigmoid fonksiyon gibi bu şartları sağlayan $arctan(x)$ fonksiyonu da logistik regresyonda sınıflandırma yapmak için kullanılabilir.

5.6.3.1 Üstünlük *(Odds)* ve üstünlük oranı *(Odds Ratio)*

Logistik regresyon için bazı önemli kavramlar vardır. Bunlar; *üstünlük (odds) ve üstünlük oranı (odds ratio)* kavramlarıdır. Üstünlük, bir olayın olma olasılığının olmama olasılığına oranıdır ve

$$O = \frac{p(x)}{1 - p(x)} \tag{5.34}$$

şeklinde ifade edilir. Bir örnek ile açıklayalım. Örneğin; Denklem (5.34)' de bir x olayının üstünlüğü olarak ifade edilen $O = 3$ olsun. O halde bu x olayının gerçekleşme olasılığı gerçekleşmeme olasılığının 3 katıdır yorumu yapılabilir.

Üstünlük oranı ise farklı iki üstünlüğün birbirine oranıdır ve

$$OR = \frac{\frac{p(x_i)}{1-p(x_i)}}{\frac{p(x_j)}{1-p(x_j)}} \tag{5.35}$$

şeklinde ifade edilir.

Denklem (5.34)' e logit dönüşüm uygulanırsa,

$$logit\, p(x) = f(x) = \ln\left(\frac{p(x)}{1-p(x)}\right) = \ln e^z = \beta_0 + \beta_1 x \tag{5.36}$$

elde edilir. Bu dönüşüm ile birlikte logistik modelimiz doğrusal bir yapıya dönüşmüş oldu. Böylece sınıflandırma yapmak için gerekli β parametlerini bulmak daha kolay olacaktır [13].

5.6.3.2 Maksimum olabilirlik yöntemi (*Maximum Likelihood Method*)

Logistik Regresyon analizinde, logistik modelin β parametreleri maksimum olabilirlik yöntemi ile bulunabilir. Bu yöntemin amacı, beklenen bağımlı/sınıf değişkenin varlığını maksimum yapan β parametrelerini tespit etmektir. Bu doğrultuda öncelikle maksimum olabilirlik fonksiyonu oluşturulmalıdır [27].

İki olası kategorili $y = \{y_1, y_2\}$ bağımlı değişken ve x_i bağımsız değişken olmak üzere (x_i, y_i) verisi için olası iki kategori sınıfı olduğundan olabilirlik fonksiyonu binom dağılım gösterecektir ve

$$ML(x) = p(x_i)^{y_i}[1 - p(x_i)]^{1-y_i} \tag{5.37}$$

şeklinde ifade edilir [25].

Denklem (5.37)' nin logaritması alalım.

$$L(\beta) = \ln[ML(\beta)] = \sum_{i=1}^{2} y_i \ln p(x) + (1 - y_i) \ln[1 - p(x)] \tag{5.38}$$

Amacımız $L(\beta)$' yı maksimize etmek olduğundan Denklem (5.38) ' in β_0 ve β_1 parametrelerine göre türevleri alınır. Türev alındıktan sonra çıkan eşitlikler aşağıdaki gibi sıfıra eşitlenir.

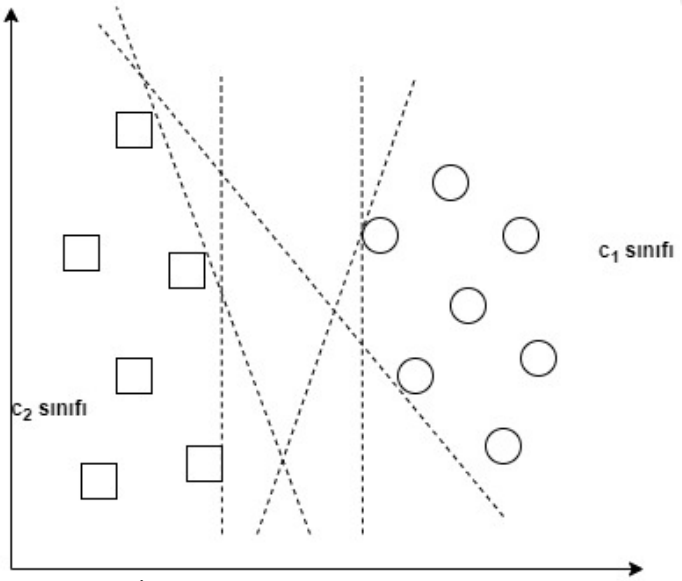
$$\frac{\partial L(\beta)}{\partial \beta_0} = \sum_{i=1}^{2} [y_i - p(x)] = 0 \tag{5.39}$$

$$\frac{\partial L(\beta)}{\partial \beta_1} = \sum_{i=1}^{2} x[y_i - p(x)] = 0 \tag{5.40}$$

Denklem (5.39) ve Denklem (5.40) kullanılarak β parametreleri bulunur.

5.7 Destek Vektör Makineleri *(Support Vector Machines (SVM))*

Destek Vektör Makineleri *(Support Vector Machines (SVM))* 1995 yılında *Corrina Cortes ve Vladimir Vapnik* tarafından geliştirilen bir sınıflandırma algoritmasıdır [7]. SVM' de temel olarak amaç; iki sınıfı, eğitim kümesi verisi ile elde edilen bir fonksiyon yardımıyla ayırmaktır. Şekil (5.11)' de bu şekilde birden fazla fonksiyon bulunabileceği resmedilmiştir. Buradaki amaç en uygun yani iki farklı sınıfı olabildiğince keskin bir şekilde ayıran fonksiyonu seçmektir. Buradaki en uygunluğun tanımı bölümün devamında yapılacaktır. Seçilen en uygun fonksiyon verilerin yer aldığı uzaya bağlı olarak bir doğru ya da bir hiper düzlem olabilir.



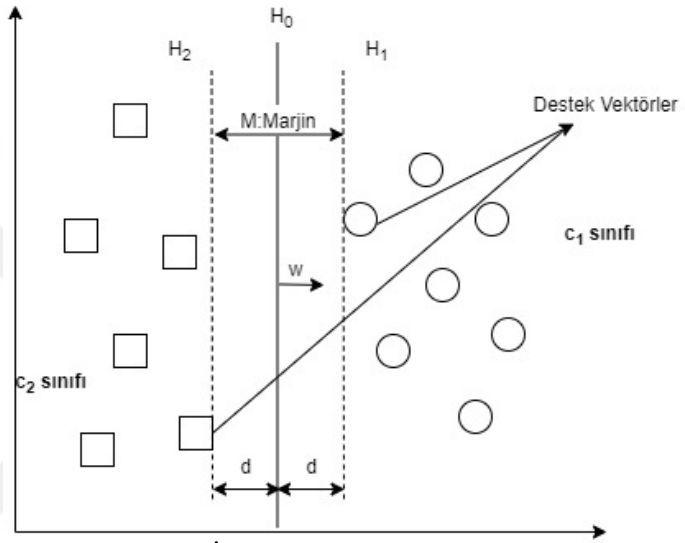
Şekil 5.11 : İki boyutlu uzayda iki sınıfı ayıran doğrular.

SVM, lineer olarak ayrılabilen ve lineer olarak ayrılamayan durumlar olarak iki ayrı alt başlıkta incelenebilir.

$D = \{x_i, c_i\} i = 1, 2, ..., n$ olacak şekilde n elemanlı bir eğitim kümesi olsun. Sınıf değişleri de $c \in \{-1, +1\}$ olarak verilsin. n boyuttaki bir uzayda verileri ayırmak için $(n-1)$ boyutlu hiper düzlemler kullanılır. Fakat biz anlatımı

kolaylaştırabilmek adına verilerin iki boyutlu uzayda dağıldığını kabul edeceğiz. İki boyutlu uzayda çalışıldığından veriler birbirinden bir doğru ile ayrılacaklardır. O halde Şekil (5.11)' de de görüldüğü gibi verileri lineer bir şekilde birbirinden ayıran birden fazla doğru olabilir.

Eğer verileri birbirinden birden fazla doğru ile ayırabiliyorsak, bu doğrulardan hangisi bizim için optimal doğru olacaktır? İşte bu soru SVM' de cevabını arayacağımız sorudur.



Şekil 5.12 : İki boyutlu uzayda sınıflandırma.

Verileri birbirinden ayıran en uygun doğru, Şekil (5.12)' de de görüldüğü gibi en yakın destek vektör ile arasındaki uzaklığın *(d)* maksimum olduğu doğrudur. Bu doğru bizim verileri sınıflandırmak için kullanabileceğimiz doğrudur ve optimal doğru olarak adlandırılır. Özet olarak amacımız Şekil (5.12)' de gösterilen *marjin* değerini maksimum yapmaktır.

n özniteliklerin sayısı, $W = \{w_1, w_2, ..., w_n\}$ ağırlık vektörü ve b bir sabit olmak üzere H_0 hiper düzlemi

$$H_0 : W^T X + b = 0 \tag{5.41}$$

şeklinde yazılabilir. O halde sırasıyla H_1 ve H_2 hiperdüzlemleri de

$$H_1 : W^T X + b = 1 \tag{5.42}$$

$$H_2 : W^T X + b = -1 \tag{5.43}$$

şeklinde ifade edilebilir. Denklem (5.11)' de ifade edilen hiper düzlemin üst tarafında (sağ) kalan veriler için

$$W^T X + b > 0, c_1 = 1 \tag{5.44}$$

eşitsizliği kullanılır. Öte yandan Denklem (5.11)' de ifade edilen hiper düzlemin alt tarafında (sol) kalan veriler için

$$W^T X + b < 0, c_2 = -1 \tag{5.45}$$

eşitsizliği kullanılır [20].

Denklem (5.44) ve (5.45) birleştirilerek

$$c_i(W^T X + b) - 1 \geq 0 \tag{5.46}$$

şeklinde tek bir eşitsizlik olarak yazılabilir.

Şekil (5.12)' de yer alan H_1 ve H_2 hiper düzlemlerini düşünelim. K hiper düzlem üzerindeki bir nokta olsun. O halde bir destek vektör ile H_0 hiper düzlemi arasındaki uzaklık;

$$d = \frac{|W^T X_K \pm b|}{||w||} = \frac{|w_1 x_{1k} + w_2 x_{2k} + ... + w_n x_{nk} + b|}{\sqrt{w_1^2 + w_2^2 + ... + w_n^2}} \tag{5.47}$$

şeklinde bulunur. O halde H_1 hiper düzlemi ile x_1 destek vektörü arasındaki uzaklık Denklem (5.42) ve Denklem (5.43)' ün yardımlarıyla

$$d = \frac{W^T x_1 + b}{||w||} = \frac{1}{||w||} \tag{5.48}$$

şeklinde ifade edilebilir [20]. Diğer hiper düzleme olan uzaklık da aynı olacağından H_1 ve H_2 hiper düzlemleri arasındaki uzaklık yani marjin

$$M = 2d = \frac{2}{||w||} \tag{5.49}$$

olacaktır. Daha önce de belirttiğimiz gibi amaçlarımızdan biri marjini maksimum yapmak olduğundan bu amaç doğrultusunda $||w||$ değerinin minimize etmemiz gerekecektir.

$$w^T w = ||w|| = ww = \sum_{i=1}^{n} w_i^2 \tag{5.50}$$

Marjini maksimum yapabilmek adına aşağıda yer alan doğrusal olmayan optimizasyon probleminin çözülmesi gerekir [20].

$$\text{Amaç:} \quad min\frac{w^T w}{2}, \quad \text{Koşul:} \quad c_i(w^T w_i + b) \geq 1 \tag{5.51}$$

Bu problem Lagrange fonksiyonu yardımıyla çözülebilir. Her α_i bir Lagrange çarpanı olmak üzere, Lagrange fonksiyonu,

$$L(w,b,\alpha) = \frac{1}{2}(w^T w) - \sum_{i=1}^{n} \alpha_i[c_i(w^T x_i + b) - 1] \tag{5.52}$$

şeklinde ifade edilir [20].

5.7.1 Karush-Kuhn-Tucker koşulları

Denklem (5.52)' de verilen Lagrange fonksiyonu $L(w,b,\alpha)$ *Karush-Kuhn-Tucker (KKT)* koşulları kullanılarak bir dual probleme dönüştürülür. Denklem (5.53) ve (5.54)' de KKT koşulları verilmiştir.

$$\frac{\partial L(w,b,\alpha)}{\partial w} = 0 \implies w^T = \sum_{i=1}^{n} \alpha_i c_i x_i \tag{5.53}$$

$$\frac{\partial L(w,b,\alpha)}{\partial b} = 0 \implies 0 = \sum_{i=1}^{n} c_i x_i \tag{5.54}$$

KKT koşulları Denklem (5.52)' de verilen $L(w,b,\alpha)$ Lagrange fonksiyonunda yerine yazılırsa;

$$
\begin{aligned}
L(w,b,\alpha) =& \frac{1}{2}(w^T w) - \sum_{i=1}^{n} \alpha_i \left[c_i(w^T x_i + b) - 1 \right] \\
=& \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j c_i c_j x_i^T x_j - \sum_{i,j=1}^{n} \alpha_i \alpha_j c_i c_j x_i^T x_j - \sum_{i=1}^{n} \alpha_i \\
L(w,b,\alpha) =& -\frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j c_i c_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i
\end{aligned}
\tag{5.55}
$$

elde edilir.

5.7.2 Dual çözüm

En uygun hiper dizlemi bulmak için Lagrange fonksiyonu $L(w,b,\alpha)$ duali $(i = 1,2,\ldots,n)$ olmak üzere $a_i \geq 0$ için maksimize etmelidir [20]. Dual model aşağıdaki gibi ifade edilebilir.

Maximum: $\alpha^T I - \frac{1}{2}\alpha^T H \alpha$

Koşul: $\alpha \geq 0$

H Hessien matrisini, I birim matrisi temsil etmektedir. O halde H matrisi

$$
H_{ij} = c_i c_j x_i x_j = c_i c_j x_i^T x_j \tag{5.56}
$$

şeklinde ifade edilebilir. Yukarıda verilen dual modelin bir (w^*, b^*, a^*) optimal noktası için Lagrange Çarpanları (a_i^*) Denklem (5.53) ve Denklem (5.54)' deki gibi optimal hiper düzlemin w^* ve b^* değişkenlerini belirler. V destek vektörleri ve $||V||$ destek vektörlerin sayısını temsil etsin. O halde

$$
w^* = \sum_{x_{i \in V}}^{n} a_i^* c_i x_i \tag{5.57}
$$

$$
b^* = \frac{1}{||V||} \left(\sum_{i=1}^{||V||} \left(\frac{1}{c_i} - x_i^T w^* \right) \right) \tag{5.58}
$$

yazılabilir [20].

Bu bilgiler ışığında bir x_i verisini SVM ile sınıflandırmak için

$$
f(x) = sgn(\sum_{i=1}^{n} \alpha_i c_i x_i^T x + b) \tag{5.59}
$$

fonksiyonu kullanılabilir. Destek vektörler cinsinden ifade edilmek istenirse,

$$f(x) = sgn(w^* x^T x + b) \tag{5.60}$$

şeklinde ifade edilebilir.

Destek Vektör Makinesi kullanılarak bir sınıflandırma yapma sürecini daha iyi anlatabilmek adına bir örnek yapalım.

- İki boyutlu uzayda destek vektör makinesi sınıflandırıcısını kullanarak *Siyah* sınıfına dahil olan $(0,0),(1,2)$ ve *Beyaz* sınıfına dahil olan $(2,2)$ noktalarını birbirinden ayıran en uygun fonksiyonu / doğruyu bulalım.

  Denklem (5.55)' de verilen Lagrange Fonksiyonu'nu

$$L(w,b,\alpha) = -\tfrac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j c_i c_j x_i x_j + \sum_{i=1}^{n} \alpha_i$$

  kullanalım. Örnekte verilen noktalardan yola çıkarak

$$x_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, x_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, x_3 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

  vektörleri yazılabilir. Sınıf vektörü ise

$$c = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}$$

  olarak yazılabilir. Lagrange Fonksiyonu'nu açık bir şekilde yazalım. $x_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ olduğundan Lagrange Fonksiyonu,

$$\begin{aligned} L(w,b,\alpha) =& \alpha_1 + \alpha_2 + \alpha_3 \\ & - \frac{1}{2}\left(\alpha_2^2 . c_2^2 x_2^T . x_2 + \alpha_2 . \alpha_3 . c_2 . c_3 . x_2^T . x_3 + \alpha_3 . \alpha_2 . c_3 . c_2 . x_3^T . x_2 + \alpha_3^2 . c_3^2 . x_3^T . x_3\right) \end{aligned}$$

  şeklinde yazılabilir. O halde

$$
\begin{aligned}
L(w,b,\alpha) =& \alpha_1+\alpha_2+\alpha_3 \\
&-\frac{1}{2}\left(\alpha_2^2.(-1)^2\begin{bmatrix}1 & 2\end{bmatrix}.\begin{bmatrix}1\\2\end{bmatrix}+\alpha_2.\alpha_3.(-1).1.\begin{bmatrix}1 & 2\end{bmatrix}.\begin{bmatrix}2\\2\end{bmatrix}\right. \\
&\left.+\alpha_3.\alpha_2.1.(-1).\begin{bmatrix}2 & 2\end{bmatrix}.\begin{bmatrix}1\\2\end{bmatrix}+\alpha_3^2.(1)^2.\begin{bmatrix}2 & 2\end{bmatrix}.\begin{bmatrix}2\\2\end{bmatrix}\right) \\
=& \alpha_1+\alpha_2+\alpha_3-\frac{1}{2}\left(5\alpha_2^2-6\alpha_2.\alpha_3-6\alpha_3.\alpha_2+8\alpha_3^2\right)
\end{aligned}
$$

Denklem (5.53)' den $\sum_{i=1}^{3} c_i.\alpha_i = 0$ olduğunu biliyoruz. Bu eşitlik yardımıyla

$$\alpha_3 = \alpha_2+\alpha_1$$

elde edilir. Bu eşitlik Lagrange Fonksiyonu' nda yerine yazılırsa

$$L(w,b,\alpha) = 2\alpha_3-\tfrac{1}{2}\left(5\alpha_2^2-12\alpha_2.\alpha_3+8\alpha_3^2\right)$$

α_2 ve α_3' ü bulmak için Lagrange Fonksiyonu' nun α_2 ve α_3' e göre türevlerini alıp sıfıra eşitleyelim.

$$\frac{\partial L(w,b,\alpha)}{\partial \alpha_2} = -5\alpha_2+6\alpha_3 = 0$$

$$\frac{\partial L(w,b,\alpha)}{\partial \alpha_3} = 4\alpha_3-3\alpha_2-1 = 0$$

Bu iki eşitlik kullanılarak $\alpha_2 = 3$ ve $\alpha_3 = \frac{5}{2}$ bulunur. Bu iki değer kullanılarak $\alpha_1 = \frac{1}{2}$ bulunur.

Tüm α değerleri bulunduğuna göre $\alpha = \begin{bmatrix}\frac{1}{2}\\3\\\frac{5}{2}\end{bmatrix}$ yazılabilir.

Denklem (5.57) ve Denklem (5.58) yardımıyla sırasıyla w^* ve b^*' ın değerlerini bulalım.

$$
\begin{aligned}
w^* =& \alpha_2 c_2 x_2+\alpha_3 c_3 x_3 \\
=& 3(-1)\begin{bmatrix}1\\2\end{bmatrix}+\frac{5}{2}(1)\begin{bmatrix}2\\2\end{bmatrix} \\
=& \begin{bmatrix}\frac{17}{4}\\\frac{13}{4}\end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
b^* &= \frac{1}{||V||}\left(\sum_{i=1}^{||V||}\left(\frac{1}{c_i} - x_i^T w^*\right)\right) \\
&= \frac{1}{2}\left(\frac{1}{-1} - \begin{bmatrix}1 & 2\end{bmatrix}\begin{bmatrix}\frac{17}{4} \\ \frac{13}{4}\end{bmatrix} + \frac{1}{1} - \begin{bmatrix}2 & 2\end{bmatrix}\begin{bmatrix}\frac{17}{4} \\ \frac{13}{4}\end{bmatrix}\right) \\
&= -\frac{103}{8}
\end{aligned}
$$

Tüm bilinmeyenler bulunduğuna göre artık sınıflandırma fonksiyonumuzu yazabiliriz. Sınıflandırılmak istenen yeni bir x_i verisi için aşağıdaki fonksiyon kullanılacaktır.

$$
\begin{aligned}
f(x) &= sgn\left(w^* x + b\right) \\
&= sgn\left(\frac{17}{4}x_1 + \frac{13}{4}x_2 - \frac{103}{8}\right)
\end{aligned}
$$

6. DENEYLER

6.1 Mushroom Veri Seti

UCI Machine Learning Repository' den alınan bu veri seti, Agaricus ve Lepiota Ailesine karşılık gelen mantar türü örneklerini içerir. Mushroom veri seti 22 adet özniteliğe sahip 8124 adet kayıttan oluşur. Bu özniteliklerin hepsi kategorik yapıdadır. Bu 8124 kayıt *class* sınıf değişkeni altında *edible* ve *poisonous* olmak üzere iki farklı kategoriye sahiptir. Her bir örnek yenilebilir *(edible)* ya da zehirli *(poisonous)* olarak tanımlanmıştır. Bu 22 adet öznitelik kullanılarak mantarın yenilebilir mi yoksa zehirli mi olduğu tahmin edilecektir.

Ayrıca bu veri seti *stalk-root* özniteliğinde eksik veriler barındırmaktadır. Buradaki öncelikli işimiz veri ön işleme kapsamında olan söz konusu eksik verileri temizlemek ve ya da uygun bir şekilde doldurmaktır. Bu işlem yapıldıktan sonra gerekli veri dönüştürme işlemleri de yapılacaktır.

Bu veri seti üzerinde tez kapsamında yer alan algoritmaları uygulayarak hangi algoritmanın ne kadar başarılı olduğunu analiz edeceğiz.

6.2 Congressional Voting Records Veri Seti

UCI Machine Learning Repository' den alınan bu veri seti, ABD Meclis Üyelerinin 16 farklı türde verdikleri oyları içermektedir. Bu oyların karşılığında da democrat (demokrat) ve repuclician (cumhuriyetçi) olmak üzere iki farklı sınıfa ayrılmaktadırlar. Bu 16 öznitelik kullanılarak meclik üyelerinin demoktrat mı yoksa cumhuriyetçi mi olduğu tahmin edilecektir. Biz bu deneyde sırasıyla K-Means, K-En Yakın Komşu, Naive Bayes, Karar Ağacı ve Destek Vektör Makineleri algoritmalarını uygulayarak verilen örnekleri sınıflandıracağız.

Congressional Voting veri seti 16 tane özniteliğe sahip 435 adet kayıtta oluşur. Bu 435 kayıt *class* sınıf değişkeni altında *democrat* ve *republician* olmak üzere iki farklı kategoriye sahiptir. Ek olarak bu veri seti her bir özniteliğinde eksik veriler

barındırmaktadır. Bu veri seti üzerinde bir sonuca varmadan önce bu eksik veriler ya temizlenmeli ya da uygun bir şekilde doldurulmalıdır.

Bu veri seti üzerinde tez kapsamında yer alan algoritmalar uygulanarak hangi algoritmanın ne kadar başarılı olduğu analiz edilecektir.

6.3 Tic-Tac-Toe Veri Seti

UCI Machine Learning Repository' den alınan bu veri seti, 958 kayıt ve 9 adet öznitelik içerir. Her bir öznitelik kategoriktir. Öznitelikte yer alan değerler sırasıyla *x*, *o* ve *b* dir. x ve o sırasıyla x ve o oyuncularının yaptığı hamleleri temsil etmektedir. *b* ise boş alanı temsil eder. Bu veri seti *x*' in ilk kez oynadığı kabul edilen tic-tac-toe oyunlarının sonunda olası tüm yerleşimleri barındırır. Bu veri setinden bulunan iki adet sınıf değişkeni bu öznitelikler sonucunda *p n* olmak üzere sırasıyla *x*' in kazanması ve *x*' in kazanamaması olarak belirtilmiştir. Söz konusu 9 öznitelik kullanılarak *x*' in kazanıp kazanamadığı tahmin edilecektir.

6.4 Veri Seti Yükleme ve Veri Ön İşleme

Mushroom Veri Seti içeri *sklearn* paketi içindeki *pandas* kütüphanesi yardımıyla *mushroom.csv* dosyası olarak alınmıştır. Aynı kütüphane kullanılarak Congressional Voting Records veri seti *voting.csv* ve Tic-Tac-Toe veri seti ise *tictactoe.csv* dosyası olarak alınmıştır. Ayrıca *numpy*, *mathplotlib* ve *seaborn* kütüphaneleri de sırasıyla gerekli matematiksel ve görselleştirme işleri için yüklenmiştir.

6.4.1 Eksik/Boş veri temizleme ve doldurma

Mushroom veri setinde *stalk-root* özniteliğinde boş *(null)* veriler vardır. Bu boş veriler söz konusu öznitelikteki frekansı en çok olan *b* değeri ile doldurulmuştur. *Congressional Voting Records* veri setinde her öznitelikte boş değerler vardır. Bu boş değerler söz konusu öznitelikteki frekansı fazla olan değerler ile doldurulmuştur. *Tic-Tac-Toe* veri setinde herhangi bir öznitelik için boş değer yoktur.

Veri setlerindeki boş veriler *.isnull().any()* komutuyla tespit edilip yerine eksik verilere karşı bir yaklaşım çeşidi olan frekansı fazla olan değer konulmuştur.

6.4.2 Veri dönüştürme

Mushroom veri setinde öznitelikler arasında herhangi bir hiyerarşi olmadığından bu öznitelikler gölge değişkenlere çevrilmiştir. *Congressional Voting Records* veri setinde öznitelikler sadece iki farklı değer içerdiğinden bu özniteliklere labelencoding uygulayarak dönüşüm gerçekleştirildi. *Tic-Tac-Toe* veri setinde yine özniteliklerin değerleri arasında bir hiyerarşi olmadığından bu öznitelikler gölge değişkenler olarak ifade edilmiştir.

Gölge değişken olarak ifade etmek için *pd.get dummies()* komutu label encoding işlemi için ise *LabelEncoder.fit transform()* komutu kullanılmıştır.

6.4.3 Eğitim ve test kümesi ayırma

Gözetimli öğrenme yöntemleri çatısı altında bulunan bir algoritma ile bir model kuruluyorsa veri setini eğitim ve test kümesi olarak ikiye ayırmak gerekliliğinden bahsetmiştik. Bu ayırma işlemini birden fazla yolla yapabiliriz. 3. Bölüm' de bahsedildiği üzere KFCV, Hold-Out, LOOCV ve Re-Substitution yöntemleri kullanılabilir. Bu tez kapsamında uygulanan deneylerde eğitim ve test kümesi ayırma işlemleri KFCV yöntemiyle yapılmış ve katman *fold* sayısı 10 olarak alınmıştır. Bu sayede modeli eğitme aşamasında modelimiz veri setindeki her bir parçayı eğitim seti olarak alabilecek. Bu sayede modelimiz daha doğru bir sonuç verecektir. Ek olarak KFCV' de seçilen katman sayısı için bir kural olmamakla beraber 10 sayısı genel olarak kabul görmüş bir değerdir.

6.5 Algoritmalar

K-Means

Veri seti üzerinde K-Means algoritmasını uygulayabilmek için *sklearn.cluster* paketinden *K-Means* sınıfı yüklendi. Fakat KMeans sınıfı bizden kümeleme yapmak için bir *K* değeri isteyeceğinden optimum *K* değerini bulmak için WCSS-Küme Sayısı grafiğini oluşturuldu ve optimum küme sayısını belirlendi. Belirlenen optimum küme sayısı ile K-Means algoritması çalıştırıldı.

KNN

Veri seti üzerinde KNN algortimasını uygulayabilmek için *sklearn.neighbors* paketinden *KNeighborsClassifier* sınıfı yüklendi. KNN bir gözetimli öğrenme

yöntemi olduğundan veri seti eğitim ve test olarak ayrıldı. Bu ayrım için *sklearn.neighbors* paketinden *train-test-split* sınıfı yüklendi. Eğitim verimizle *.fit()* komutuyla modeli eğitirken test verisi ile de test verisi sınıflandırıldı. Bu işlemi yapmadan önce bir *for* döngüsü yardımıyla optimum *K* değeri bulundu.

Karar Ağaçları

Karar ağaçları algoritması için de *DecisionTreeClassifier* sınıfı yüklendi.Gözetimli öğrenme yönteminin gereği olarak veri kümesi eğitim ve test oalrak ikiye ayrıldı. Burada önemli olan noktalardan biri karar ağacının ağaç yapısının dallanma kriterini belirlemektir. Burada dallanma kriteri olarak *Entropi* yöntemi kullanılmıştır. Yapılan sınıflandırma sonucunda aşağıdaki performans ölçüt değerleri elde edilmiştir.

Naive Bayes

Naive Bayes ile sınıflandırma yapabilmek adına *GaussianNB* sınıfı yüklendi. Eğitim ve test kümeleri oluşturulduktan sonra model eğitilip test verisi ile sınıflandırmalar yapıldı.

Logistik Regresyon

Veri seti üzerinde Logistik Regresyon algoritmasını uygulamak için *sklearn.linear model* altında yer alan *LogisticRegression* sınıfı yüklendi. Yine gözetimli öğrenme yönteminin gereği olarak veri kümesi eğitim ve test olarak ikiye ayrıldı. Bu algoritma bize çıktı olarak bir kaydın bir sınıfa ait olma olasılığını verdiğinden, algoritma yazilirken bir threshold değeri belirlendi. Sınıflandırma sonucunda çıkan olasılık thresholdun ilgili yerindeki sınıfa adreslendi.Veri setleri iki olası kategoriye sahip olduğundan *LogisticRegression* sınıfına ait *multi-class* parametresi boş bırakıldı. Ayrıca *Mushroom* veri seti nispeten orta büyüklükte bir seti olduğundan *solver* parametresi olarak *saga* seçildi. Diğer iki veri seti nispeten küçük veri setleri olduğundan bu parametre *liblinear* olarak belirtildi.

SVM

Veri seti üzerinde destek vektör makinesi algorimasını uygulayabilmek için *svm* sınıfı yüklendi. Veri kümesi eğitim ve test olarak ayrıldıktan sonra sınıflandırma

işlemi *LinearSVM* yöntemiyle yapıldı. Yapılan sınıflandırma sonucunda elde edilen performans ölçüt değerleri aşağıdadır.

6.6 Model Değerlendirme ve Cohen Kappa Skorları

Veri setlerinde uygulanan her bir modelin başarı metrikleri ayrı ayrı hesaplanmıştır. Modellerin yaptıkları tahminlerin başarısını daha sağlıklı inceleyebilmek adına Cohen Kappa Skorları hesaplanmıştır. Böylece bir modelin doğruluğunun ne kadar şansa bağlı olup olmadığı gözlemlenebilecektir.

7. SONUÇLAR VE ANALİZ

Çizelge (7.1)' de her bir veri seti üzerinde uygulanan algoritmaların söz konusu veri seti üzerinde sergilemiş oldukları model performans ölçütleri hesaplanmıştır.

Çizelge 7.1 : Veri setleri için modellerin ortalama performans skorları.

Veri Seti	Algoritma	Doğruluk	Kesinlik	Duyarlılık	F-Ölçütü	Kappa
Mushrooms	**K-Means**	0.89	0.98	0.78	0.87	0.78
	KNN	0.98	0.98	0.95	0.96	0.78
	Karar Agaci	0.97	0.93	0.94	0.93	0.97
	Naive Bayes	0.96	0.93	0.91	0.92	0.94
	Logistik Reg.	0.98	0.97	0.95	0.96	0.93
	SVM	0.98	0.97	0.95	0.96	0.96
Congressional Voting Records	**K-Means**	0.88	0.79	0.93	0.85	0.75
	KNN	0.92	0.90	0.92	0.90	0.75
	Karar Agaci	0.94	0.92	0.92	0.93	0.84
	Naive Bayes	0.93	0.90	0.93	0.91	0.88
	Logistik Reg.	0.95	0.94	0.95	0.94	0.85
	SVM	0.96	0.94	0.95	0.95	0.90
Tic-Tac-Toe	**K-Means**	0.57	0.69	0.64	0.66	0.098
	KNN	0.65	0.66	0.58	0.60	0.098
	Karar Agaci	0.59	0.65	0.49	0.52	0.17
	Naive Bayes	0.40	0.65	0.31	0.39	0.12
	Logistik Reg.	0.83	0.69	0.68	0.68	0.60
	SVM	0.98	0.70	0.70	0.70	0.76

Bu ölçütler gözetimli öğrenme modelleri için 10 katlı çapraz geçerleme sonucunda elde edilen ortalama değerlerdir. Söz konusu gözetimli öğrenme modellerinin 10 katlı çapraz geçerleme sürecinin her bir iterasyonunda göstermiş oldukları değerler Şekil (7.1)' de verilmiştir.

Görüleceği üzere her üç veri setinde de bariz bir şekilde en yüksek başarı değerleri SVM tarafından elde edilmiştir.

Mushroom veri seti özelinde bir değerlendirme yapalım. Bu veri setinde modellerin doğruluk ve F-Ölçütü değerleri birbirlerine yakın değerler olarak görülmektedir. Bu da bize modellerin bu veri seti üzerinde iyi bir performans gösterdiklerini göstermektedir. Fakat aynı anda doğruluk değerinin yüksek olup buradakinin

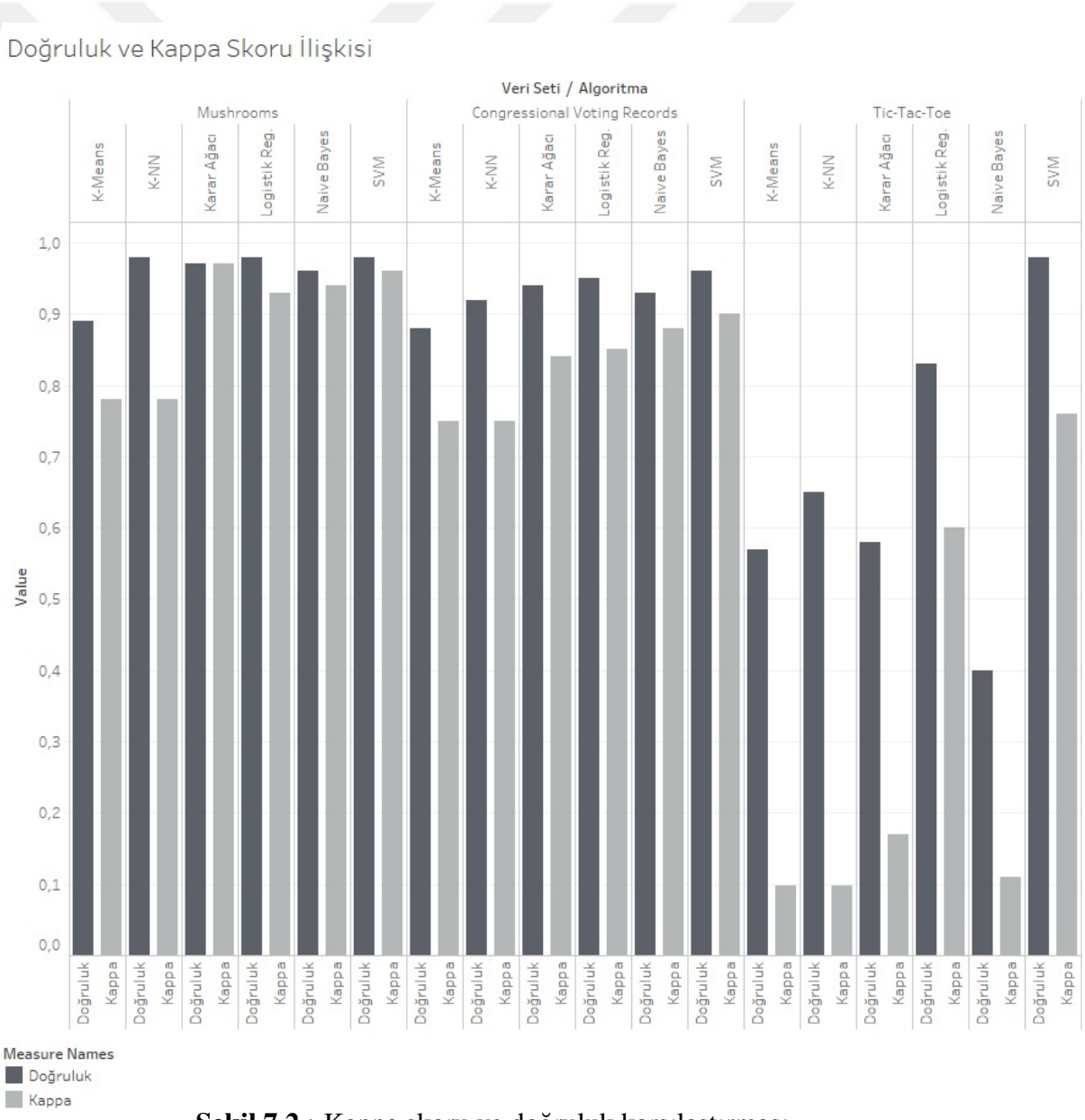
			Çapraz Doğrulama İterasyon Adımı									
Veri Seti	Algoritma	Metrik	1	2	3	4	5	6	7	8	9	10
Mushrooms	KNN	Doğruluk	0.91	1	1	1	1	0.99	0.98	0.95	1	0.99
	KNN	Kesinlik	0.90	1	1	1	1	1	0.99	0.95	1	0.99
	KNN	Duyarlılık	0.55	1	1	1	1	0.99	0.98	1	1	1
	KNN	F-Ölçütü	0.69	1	1	1	1	0.99	0.99	0.97	1	0.99
	Karar Agaci	Doğruluk	0.91	0.94	0.98	1	0.99	0.98	0.93	0.95	1	1
	Karar Agaci	Kesinlik	0.90	0.69	0.84	1	1	1	0.99	0.95	1	1
	Karar Agaci	Duyarlılık	0.55	0.98	0.98	1	0.99	0.98	0.93	0.99	1	1
	Karar Agaci	F-Ölçütü	0.69	0.82	0.89	1	0.99	0.99	0.96	0.97	1	1
	Naive Bayes	Doğruluk	0.86	0.99	0.96	0.99	1	0.99	0.93	0.95	0.99	0.98
	Naive Bayes	Kesinlik	0.59	0.94	0.91	0.99	1	0.99	0.94	0.94	0.99	0.97
	Naive Bayes	Duyarlılık	0.58	1	0.63	1	1	0.99	0.97	1	1	1
	Naive Bayes	F-Ölçütü	0.59	0.97	0.74	0.99	1	0.99	0.96	0.97	0.99	0.98
	Logistik Reg.	Doğruluk	0.91	0.99	1	1	1	0.99	0.97	0.95	1	1
	Logistik Reg.	Kesinlik	0.90	0.94	1	1	1	1	0.99	0.95	1	1
	Logistik Reg.	Duyarlılık	0.55	1	1	1	1	0.99	0.97	1	1	1
	Logistik Reg.	F-Ölçütü	0.69	0.97	1	1	1	0.99	0.98	0.97	1	1
	SVM	Doğruluk	0.91	0.99	1	1	1	0.99	0.98	0.95	1	1
	SVM	Kesinlik	0.90	0.94	1	1	1	1	0.99	0.95	1	1
	SVM	Duyarlılık	0.55	1	1	1	1	0.99	0.98	1	1	1
	SVM	F-Ölçütü	0.69	0.96	1	1	1	0.99	0.99	0.97	1	1
Congressional Voting Records	KNN	Doğruluk	0.97	0.93	0.97	0.86	0.95	0.95	0.93	0.97	0.81	0.88
	KNN	Kesinlik	0.94	0.90	1	0.78	0.91	1	0.94	0.94	0.65	0.94
	KNN	Duyarlılık	1	0.95	0.92	0.94	0.91	0.89	0.88	1	0.92	0.8
	KNN	F-Ölçütü	0.96	0.92	0.96	0.85	0.91	0.94	0.91	0.96	0.74	0.86
	Karar Agaci	Doğruluk	0.95	0.97	0.95	0.93	0.97	0.95	0.95	0.88	0.93	0.95
	Karar Agaci	Kesinlik	0.93	1	0.92	0.90	0.92	1	1	0.93	0.76	0.94
	Karar Agaci	Duyarlılık	0.93	0.95	0.92	1	1	0.89	0.88	0.81	1	0.95
	Karar Agaci	F-Ölçütü	0.93	0.97	0.92	0.95	0.96	0.91	0.94	0.93	0.83	0.92
	Naive Bayes	Doğruluk	0.97	0.93	0.97	0.81	0.95	0.95	0.97	0.97	0.81	0.93
	Naive Bayes	Kesinlik	0.94	0.90	1	0.73	0.91	1	1	0.94	0.65	0.94
	Naive Bayes	Duyarlılık	1	0.95	0.92	0.89	0.91	0.89	0.94	1	0.92	0.9
	Naive Bayes	F-Ölçütü	0.96	0.92	0.96	0.80	0.91	0.94	0.97	0.96	0.76	0.92
	Logistik Reg.	Doğruluk	0.97	0.93	0.97	0.93	1	0.95	0.97	0.97	0.88	0.95
	Logistik Reg.	Kesinlik	0.94	0.90	1	0.90	1	1	1	0.94	0.76	0.95
	Logistik Reg.	Duyarlılık	1	0.95	0.92	0.94	1	0.89	0.94	1	0.92	0.95
	Logistik Reg.	F-Ölçütü	0.96	0.92		0.96	0.92	1	0.94	0.97	0.96	0.83
	SVM	Doğruluk	0.97	0.95	0.97	0.95	1	0.95	0.97	0.97	0.90	0.93
	SVM	Kesinlik	0.94	0.95	1	0.90	1	1	1	0.94	0.81	0.94
	SVM	Duyarlılık	1	0.95	0.92	1	1	0.89	0.94	1	0.92	0.90
	SVM	F-Ölçütü	0.96	0.95	0.96	0.95	1	0.94	0.97	0.96	0.86	0.92
Tic-Tac-Toe	KNN	Doğruluk	0.52	0.96	0.93	0.83	0.83	0.82	0.66	0.26	0.17	0.53
	KNN	Kesinlik	1	1	1	1	1	1	0.62	0	0	0
	KNN	Duyarlılık	0.52	0.96	0.93	0.83	0.83	0.82	0.92	0	0	0
	KNN	F-Ölçütü	0.68	0.98	0.96	0.90	0.90	0.90	0.74	0	0	0
	Karar Agaci	Doğruluk	0.46	0.55	1	0.23	0.84	0.77	0.64	0.48	0.58	0.2
	Karar Agaci	Kesinlik	1	1	1	1	1	1	0.60	0	0	0
	Karar Agaci	Duyarlılık	0.46	0.57	1	0.23	0.82	0.85	0.98	0	0	0
	Karar Agaci	F-Ölçütü	0.63	0.72	1	0.38	0.92	0.87	0.70	0	0	0
	Naive Bayes	Doğruluk	0.23	0.73	0.65	0.31	0.40	0.20	0.48	0.28	0.29	0.33
	Naive Bayes	Kesinlik	1	1	1	1	1	1	0.50	0	0	0
	Naive Bayes	Duyarlılık	0.23	0.73	0.65	0.31	0.40	0.20	0.58	0	0	0
	Naive Bayes	F-Ölçütü	0.38	0.85	0.79	0.47	0.57	0.34	0.54	0	0	0
	Logistik Reg.	Doğruluk	0.89	0.98	1	0.92	1	1	0.98	0.69	0.66	0.17
	Logistik Reg.	Kesinlik	1	1	1	1	1	1	0.98	0	0	0
	Logistik Reg.	Duyarlılık	0.89	0.98	1	0.92	1	1	1	0	0	0
	Logistik Reg.	F-Ölçütü	0.94	0.99	1	0.96	1	1	0.99	0	0	0
	SVM	Doğruluk	1	1	1	1	1	1	1	1	1	0.83
	SVM	Kesinlik	1	1	1	1	1	1	1	0	0	0
	SVM	Duyarlılık	1	1	1	1	1	1	1	0	0	0
	SVM	F-Ölçütü	1	1	1	1	1	1	1	0	0	0

Şekil 7.1 : Gözetimli öğrenme modellerinin KFCV skorları.

aksine F-Ölçütü değeri düşük olsaydı bu veri seti için sınıflarının dağılım frekansının dengesiz olabileceğinden bahsedebilirdik. Kappa skorları için ise her bir modelin gerçek değerler ile sınıflandırdığı değerler arasındaki uyumun yüksek olduğu görülmektedir. Bu da bu modellere güvenebileceğimizi göstermektedir.

Congressional Voting Records veri seti için de benzer şeyler söylenebilir. Yüksek doğruluk değerleri ile birlikte gelene F-ölçütleri modelin iyi bir performansa sahip olduğunu ve veri setinin sınıflarının dağılım frekansının düzenli olduğunu söylemektedir.

Tic-Tac-Toe veri setine geldiğimizde burada durumun biraz farklı olduğu gözükmektedir. Modellerin performansı bu veri setinde genel olarak düşmüştür. SVM için konuşacak olursak; her ne kadar iyi bir sınıflandırma başarısına sahip olsa da kesinlik ve doğruluk değerleri bu doğruluk değeriyle örtüşmeyecek düzeyde kötüdür. Ayrıca F- Ölçütünün doğruluk değerinden bu denli farklı olması bu veri setinin dengesiz bir veri seti olduğunu söylemektedir.



Şekil 7.2 : Kappa skoru ve doğruluk karşılaştırması.

Kod kısmına bakıldığında gerçekten Tic-Tac-Toe verisindeki sınıflarının dağılım frekan oranı yaklaşık 1:2 şeklindedir.

Şekil (14.2)' de doğruluk ile kappa skoru arasındaki farklılaşmalar gösterilmiştir. Bu grafikte yer alan kappa skorları Çizelge (6.5) yardımıyla yorumlanabilir. Tic-Tac-Toe veri setinde KNN ile karar ağacın sınıflandırma başarılarına bakacak olursak, KNN' in karar ağacına göre daha iyi bir doğruluk yakaladığını görülürken kappa skoru olarak daha düşük bir değer ürettiğini görmekteyiz. Bu da demek oluyor ki, KNN' in tahmin ettiği sınıflar ile gerçek sınıflar arasındaki uyumun şansa bağlı olma olasılığı karar ağacına göre daha fazladır. Kappa skorunun 0.4' den az olması makul bir uyumu ifade etmez. Özellikle bu denli birbirine yakın doğruluk değeri üreten modeller arasında seçim yaparken kappa skorunun mutlaka göz önüne alınması gerekir. Her ne kadar doğruluk değeri bize modelin başarısı hakkında genel bir çerçeve çizse de bu doğruluk değerinin şansa bağlı olup olmadığını ölçmek de bir o kadar değerlidir ve model seçimimizi doğrudan etkileyebilir.

KAYNAKLAR

[1] **Landis, J. ve Koch, G.** (1977). The Measurement of Observer Agreement for Categorical Data.

[2] **Liew, L.**, What is Overfitting in Trading?, `https://algotrading101.com/learn/what-is-overfitting-in-trading/`.

[3] **Swapna, T. ve Sravani, Y.** (2019). Performance Analysis of Classification Algorithms on Parkinson's Dataset with Voice Attributes.

[4] **Pahva, P.**, **Manju, P. ve Renu, M.** (2014). Performance Analysis of Classification Algorithms.

[5] **Fix, E. ve Hodges, L.** (1951). An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation.

[6] **Hastie, T.**, **Tibshirani, R. ve Friedman, J.** (2009). *The elements of statistical learning*, Springer Series in Statistics, Springer, New York, second sürüm, `https://doi.org/10.1007/978-0-387-84858-7`, data mining, inference, and prediction.

[7] **Cortes, C. ve Vapnik, V.** (1995). Support Vector Networks.

[8] **Bishop, C.M.** (2006). *Pattern recognition and machine learning*, Information Science and Statistics, Springer, New York, `https://doi.org/10.1007/978-0-387-45528-0`.

[9] **Alpaydin, E.** (2010). *Introduction to Machine Learning*, The MIT Press, 2nd sürüm.

[10] **Agresti, A.** (2019). *An introduction to categorical data analysis*, Wiley Series in Probability and Statistics, John Wiley & Sons, Inc., Hoboken, NJ, third edition of [MR1394195].

[11] **Han, J.**, **Pei, J. ve Kamber, M.** (2011). *Data mining: Concepts and Techniques*.

[12] **Bilgin, M.** (2018). *Makine Öğrenmesi, Makine Öğrenmesi Teorisi ve Algoritmaları*.

[13] **Kartal, E.**, (2015), Sınıflandırmaya Dayalı Makine Öğrenmesi Teknikleri ve Kardiyolojik Risk Değerlendirmesine İlişkin Bir Uygulama.

[14] **Cohen, J.** (1960). A Coefficient of Agreement for Nominal Scales.

[15] **MacQuenn, J.** (1967). Some Methods for Classification and Analysis of Multivariate Observations.

[16] **Steinbach, M.** (2000). A Comparison of Document Clustering Techniques.

[17] **Kaygun, A.**, Veri Biliminde İstatistik ve Makine Öğrenmesi Metotları, `https://web.itu.edu.tr/kaygun/Programming/ml.html`.

[18] **Akpınar, H.** (2017). *Data: Veri Madenciliği Veri analizi*.

[19] **Guo, G.** (2003). KNN model-based approach in classification, *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", page: 986-996*.

[20] **Özkan, Y.** (2018). *Veri Madenciliği Yöntemleri*.

[21] **Muhammed, J.** (2014). *Data Mining and Analysis Fundamental Concepts and Algorithms*.

[22] **Quinlan, J.** (1986). Induction of Decision Tress.

[23] **Balaban, M. ve Kartal, E.** (2018). *Veri Madenciliği ve Makine Öğrenmesi*.

[24] **Cox, D.** (1958). The Regression of Binary Sequences.

[25] **David, W. ve Lemeshow, S.** (2000). *Applied Logistic Regression*.

[26] **Bayram, N.** (2011). Multinomial Lojistik Regresyon Analizinin İstihdamdaki İşgücüne Uygulanması.

[27] **Powers, D. ve Xie, Y.** (2000). *Statistical Methods for Categorical Data Analysis*.

EKLER

EK A : Mushrooms Veri Seti Sınıflandırma

EK B : Congressional Voting Veri Seti Sınıflandırma

EK C : Tic-Tac-Toe Veri Seti Sınıflandırma

EK A Mushroom Veri Seti Sınıflandırma

```
# Baslangic icin gerekli olan kutuphaneleri alindi.
import numpy as np  # Her turlu matematiksel islemleri yapmamiza olanak saglayan
# kutuphane.
import pandas as pd # Veri seti uzerinde islemler(okuma, yazma,...) yapma
# ve dataframe olu turmak i in.
import matplotlib.pyplot as plt # Gerekli gorsellestirme icin.

# Mushrooms veri seti yuklendi.

mushrooms = pd.read_excel ('mushroom.xlsx')

mushrooms.isnull().any() # stalk-root ozniteliginde bos degerler var.

# oznitelikler ve siniflar ayrildi.
x = mushrooms.drop(columns='class')
y = mushrooms['class']

# stalk-root ozniteligindeki bos degerler stalk-root oznitelig inde
# frekansi en yuksek olan degerle dolduruldu.

x['stalk-root'] = x['stalk-root'].fillna('b')

from sklearn.preprocessing import LabelEncoder
le_encoder = LabelEncoder()

x = pd.get_dummies(x) # golge degisken olarak donusturuldu.

y = le_encoder.fit_transform(y) # labelencoding yapildi.

# K-Means sinifini yuklendi.

from sklearn.cluster import KMeans
# KMeans algoritmas nda daha  nce  de anlatildigi gibi oncelikle
# "optimum K degeri" belirlenmesi gerekir.
# Bunun icin Dirsek Y ntemi (The Elbow Method) kullancagiz.

WCSS =[] # bos bir liste olusturuldu. Her bir K degeri icin bulunan WCSS
# degerleri bu listeye atilacak.

for K in range(1,11):
    KM = KMeans(n_clusters = K, init= 'k-means++', max_iter =300, n_init = 10)
    KM.fit(x.values) # Daha once belirledigimiz x parametresini
     # algoritmaya veriyoruz.
    WCSS.append(KM.inertia_)

# KMeans algoritmasinda en buyuk problemlerden biri rastgele
# baslangic noktasi sorunudur.
# k-means++ parametresi bizi bu durumdan kurtaracak sekilde baslangic noktalari
# seciyor.

# Dirsek metodunun grafigini gorelim.

plt.plot(range(1,11),WCSS)
plt.title ('Mushrooms veri seti icin Dirsek Y ntemi')
plt.xlabel ('Kume Sayisi')
plt.ylabel ('WCSS')
plt.show()

np.diff(WCSS)
# cikan farklara baktigimizda kume sayisinin 2'den 3' e cikarildiginda WCSS
# degerinde keskin bir azalis olmamaya
# basliyor. Dolayisiyla K=2 bizim icin optimum K degeri olarak degerlendirilebilir.

model_predicted = ['y_KM','y_KNN','y_DT','y_NB','y_LR','y_SVM']

# O halde K=2 icin K-Means uygulayalim.
```

```
KM = KMeans(n_clusters = 2, init= 'k-means++', max_iter =300, n_init = 10)
model_predicted[0]= KM.fit_predict(x)

# Cross Validation icin gerekli kutuphaneleri yuklendi.

from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
from sklearn.metrics import cohen_kappa_score,confusion_matrix
from sklearn.model_selection import KFold,cross_val_score,cross_val_predict

# 10-fold cross validation yapacagimiz icin KFold fonksiyonuna gerekli
# parametreler verildi.

KFCV = KFold(n_splits = 10, random_state = 42)

# K En Yakin Komsu sinifini Olusturalim.
# Fakat Once optimum K degerini belirleyelim.

from sklearn.neighbors import KNeighborsClassifier

values =[]

# Bu for dongusunde her bir K degeri icin KNN calisacak
# ve en yuksek dogrulugu olani alinacak.

for K in range(1,20):
    KNN=KNeighborsClassifier(n_neighbors=K,metric='euclidean')
    accuracies=cross_val_score(estimator=KNN,X=x,y=y,cv=KFCV,scoring='accuracy').mean()
    values.append(accuracies)

plt.plot(range(1,20),values)
plt.ylabel('Dogruluk')
plt.xlabel('K Sayisi')

print('En yuksek dogruluga sahip K degeri:', values.index(max(values)),',','Dogruluk:', max(values))

# Bulunan optimum K degeri ile model calistirilir.

KNN = KNeighborsClassifier(n_neighbors=values.index(max(values)),metric='euclidean')

#Karar Agaci sinifini olusturalim. Dallanma kriteri olarak Entropi kullanildi.

from sklearn.tree import DecisionTreeClassifier

DT = DecisionTreeClassifier(criterion = 'entropy')

#Naive Bayes sinifini olusturalim.

from sklearn.naive_bayes import GaussianNB

NB = GaussianNB()

#Logistik Regresyon sinifini olusturalim.

from sklearn.linear_model import LogisticRegression

LogR = LogisticRegression(solver = 'saga') # Veri seti buyuk bir veri seti
# olmadigindan solver icin 'saga' iyi bir secenek.
#Ayrica veri seti iki sinifli oldugundan multi_class ozelligi kullanilmadi.

#SVM icin svm sinifi olusturuldu.

from sklearn import svm
L_SVM = svm.LinearSVC()  # Lineer SVM' i kullanalim.

# Kullanacagimiz metrikler icin bir dizi olusturuldu.

scoring = ['accuracy' ,'precision','recall','f1']

model_estimator_list=[KNN,DT,NB,LogR,L_SVM]
model_name_list=['K-Nearest Neighbors','Decision Tree','Naive Bayes','Logistic Regression','SVM']


# Modellerin hata matrislerine bakalim.
```

```
print('K-Means icin hata matrisi: \n' ,confusion_matrix(y,model_predicted[0]))

for i in range(1,6):
  model_predicted[i]=cross_val_predict(model_estimator_list[i-1],x,y,cv=KFCV)
  print(model_name_list[i-1],'icin hata matrisi: \n',confusion_matrix(y,model_predicted[i]))


# Siniflandirma sonrasinda elde edilen metriklere bakalim.

# KMeans' in ayri degerlendirilmesinin sebebi uzerinde
# cross validation uygulanmamis olmasi.

print('K-Means modeli icin accuracy', accuracy_score(y,model_predicted[0]))
print('K-Means modeli icin precision', precision_score(y,model_predicted[0]))
print('K-Means modeli icin recall', recall_score(y,model_predicted[0]))
print('K-Means modeli icin f1', f1_score(y,model_predicted[0]))


for i in range(0,5):
    for j in range(0,4):
      metrics_scores=cross_val_score(estimator=model_estimator_list[i],
                                     X=x,y=y,cv=KFCV,scoring=scoring[j])
      print(model_name_list[i],'modeli icin ',scoring[j],metrics_scores))
      print(model_name_list[i],'modeli icin total',scoring[j],metrics_scores.mean())

# Her bir modelin Cohen Kappa Skorlarini hesaplayalim.

print('K-Means modeli icin kappa',cohen_kappa_score(y,model_predicted[0]))
for i in range(0,5):
   print(model_name_list[i],'icin kappa',cohen_kappa_score(y,model_predicted[i]))
```

EK B Congressional Voting Records Veri Seti Sınıflandırma

```
# Baslangicta gerekli olan kutuphaneler alindi.
import numpy as np  # Her turlu matematiksel islemleri yapmamiza olanak
# saglayan kutuphane.
import pandas as pd # Veri seti uzerinde islemler(okuma, yazma,...) yapma ve
# dataframe olu turmak i in.
import matplotlib.pyplot as plt # Gerekli gorsellestirme icin.

# Congressional Voting Record veri seti yuklendi.

voting = pd.read_csv ('congressional_voting.csv')

voting.isnull().any() # Her oznitelikte bos degerler mevcut.

# Oznitelikler ve siniflar ayrildi.

x = voting.drop(columns='Class')
y = voting['Class']

# Her bir oznitelikteki degerlerin frekanslarini belirleyerek null degerleri
# en cok frekansa sahip degerle dolduruldu.

for i in x.columns:
    print(x[i].value_counts('n'))


x['handicapped-infants']=x['handicapped-infants'].fillna('n')
x['water-project-cost-sharing']=x['water-project-cost-sharing'].fillna('y')
x['adoption-of-the-budget-resolution']
=x['adoption-of-the-budget-resolution'].fillna('y')
x['physician-fee-freeze']=x['physician-fee-freeze'].fillna('n')
x['el-salvador-aid']=x['el-salvador-aid'].fillna('y')
x['religious-groups-in-schools']=x['religious-groups-in-schools'].fillna('y')
x['anti-satellite-test-ban']=x['anti-satellite-test-ban'].fillna('y')
x['aid-to-nicaraguan-contras']=x['aid-to-nicaraguan-contras'].fillna('y')
x['mx-missile']=x['mx-missile'].fillna('y')
x['immigration']=x['immigration'].fillna('y')
x['synfuels-corporation-cutback']=x['synfuels-corporation-cutback'].fillna('n')
x['education-spending']=x['education-spending'].fillna('n')
x['superfund-right-to-sue']=x['superfund-right-to-sue'].fillna('y')
x['crime']=x['crime'].fillna('y')
x['duty-free-exports']=x['duty-free-exports'].fillna('n')
x['superfund-right-to-sue']=x['superfund-right-to-sue'].fillna('y')
x['export-administration-act-south-africa']
=x['export-administration-act-south-africa'].fillna('y')


# Kategorik verilerin donusumu yapildi.

from sklearn.preprocessing import LabelEncoder
le_encoder = LabelEncoder()

for i in x.columns:
    x[i] = le_encoder.fit_transform(x[i])

y = le_encoder.fit_transform(y)

# K-Means sinifi yuklendi.

from sklearn.cluster import KMeans
# KMeans algoritmas nda daha  nce  de anlatildigi gibi oncelikle "optimum K degeri"
# belirlenmesi gerekir. Bunun icin Dirsek Y ntemi (The Elbow Method) kullancagiz.

WCSS =[] # bos bir liste olusturuldu. Her bir K degeri icin bulunan WCSS
# degerleri bu listeye atildi.
```

```
for K in range(1,11):
    KM = KMeans(n_clusters = K, init= 'k-means++', max_iter =300, n_init = 10)
    KM.fit(x.values) # Daha once belirlenen x parametresini algoritmaya veriyoruz.
    WCSS.append(KM.inertia_)

# KMeans algoritmasinda en buyuk problemlerden biri
# rastgele baslangic noktasi sorunudur.
# k-means++ parametresi bizi bu durumdan kurtaracak sekilde
# baslangic noktalari seciyor.

# Dirsek metodunun grafigini gorelim.

plt.plot(range(1,11),WCSS)
plt.title ('Congressional Voting Record veri seti icin Dirsek Yontemi')
plt.xlabel ('Kume Sayisi')
plt.ylabel ('WCSS')
plt.show()

np.diff(WCSS)
# cikan farklara baktigimizda kume sayisinin 2'den 3' e cikarildiginda WCSS degerinde
# keskin bir azalis olmamaya basliyor. Dolayisiyla K=2
# bizim icin optimum K degeri olarak degerlendirilebilir.

model_predicted = ['y_KM','y_KNN','y_DT','y_NB','y_LR','y_SVM']

# O halde K=2 icin K-Means uygulayalim.
KM = KMeans(n_clusters = 2, init= 'k-means++', max_iter =300, n_init = 10)
model_predicted[0]= KM.fit_predict(x)

# Cross Validation icin gerekli kutuphaneleri yuklendi.

from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
from sklearn.metrics import cohen_kappa_score,confusion_matrix
from sklearn.model_selection import KFold,cross_val_score,cross_val_predict

# 10-fold cross validation yapacagimiz icin KFold fonksiyonuna
# gerekli parametreler verildi.
KFCV = KFold(n_splits = 10, random_state = 42)

# K En Yakin Komsu sinifini Olusturalim.
# Fakat Once optimum K degerini belirleyelim.

from sklearn.neighbors import KNeighborsClassifier

values =[]

# Bu for dongusunde her bir K degeri icin KNN calisacak
# ve en yuksek dogrulugu olani alinacak.

for K in range(1,20):
    KNN = KNeighborsClassifier(n_neighbors=K,metric='euclidea')
    accuracies = cross_val_score(estimator=KNN,X=x,y=y,cv=KFCV,scoring='accuracy').mean()
    values.append(accuracies)

plt.plot(range(1,20),values)
plt.ylabel('Dogruluk')
plt.xlabel('K Sayisi')

print('En iyi dogruluga sahip K degeri:',values.index(max(values)),',','Dogruluk:',max(values))

# Bulunan optimum K degeri ile model calistirilir.

KNN=KNeighborsClassifier(n_neighbors=values.index(max(values)),metric='euclidean')

#Karar Agaci sinifini olusturalim. Dallanma kriteri olarak Entropi kullanalim.

from sklearn.tree import DecisionTreeClassifier

DT = DecisionTreeClassifier(criterion = 'entropy')

#Naive Bayes sinifini olusturalim.
from sklearn.naive_bayes import GaussianNB

NB = GaussianNB()
```

```
#Logistik Regresyon sinifini olusturalim.
from sklearn.linear_model import LogisticRegression

LogR = LogisticRegression(solver = 'liblinear') # Veri setimiz buyuk bir
# veri seti olmadigindan solver icin 'liblinear' iyi bir secenek.
# Daha buyuk veri setlerinde 'sag' veya 'saga' secilebilir.
#Ayr ca veri setimiz iki sinifli oldugundan multi_class ozelligini kullanmadik.

#SVM icin svm sinifi olusturuldu.

from sklearn import svm
L_SVM = svm.LinearSVC()  # Lineer SVM' i kullanalim.

# Kullanacagimiz metrikler icin bir dizi olusturuldu.
scoring = ['accuracy' ,'precision','recall','f1']


model_estimator_list=[KNN,DT,NB,LogR,L_SVM]
model_name_list=['K-Nearest Neighbour',
'Decision Tree', 'Naive Bayes', 'Logistic Regression', 'SVM']


# Modellerin hata matrislerine bakalim.

print('K-Means icin hata matrisi: \n' ,confusion_matrix(y,model_predicted[0]))

for i in range(1,6):
    model_predicted[i] = cross_val_predict(model_estimator_list[i-1],x,y,cv=KFCV)
    print(model_name_list[i-1],'icin hata matrisi: \n',
          confusion_matrix(y,model_predicted[i]))


# Siniflandirma sonrasinda elde edilen metriklere bakalim.

# KMeans' in ayri degerlendirilmesinin sebebi uzerinde cross validation
# uygulanmamis olmasi.

print('K-Means modeli icin accuracy', accuracy_score(y,model_predicted[0]))
print('K-Means modeli icin precision', precision_score(y,model_predicted[0]))
print('K-Means modeli icin recall', recall_score(y,model_predicted[0]))
print('K-Means modeli icin f1', f1_score(y,model_predicted[0]))


for i in range(0,5):
    for j in range(0,4):
        metrics_scores=cross_val_score(estimator=model_estimator_list[i],X=x,y=y,
                                       cv=KFCV,scoring=scoring[j])
        print(model_name_list[i] ,'modeli icin', scoring[j], metrics_scores.mean())

# Her bir modelin Cohen Kappa Skorlarini hesaplayalim.

print('K-Means modeli icin kappa',cohen_kappa_score(y,model_predicted[0]))
for i in range(0,5):
   print(model_name_list[i],'icin kappa',cohen_kappa_score(y,model_predicted[i]))
```

EK C Tic-Tac-Toe Veri Seti Sınıflandırma

```
# Baslangicta gerekli olan kutuphaneler alindi.
import numpy as np  # Her turlu matematiksel islemleri yapmamiza olanak
# saglayan kutuphane.
import pandas as pd # Veri seti uzerinde islemler(okuma, yazma,...) yapma ve
# dataframe olu turmak i in.
import matplotlib.pyplot as plt # Gerekli gorsellestirme icin.

# Tic Tac Toe veri seti yuklendi.

ttt = pd.read_csv ('tictactoe.csv') # Tic-Tac-Toe veri seti yuklendi.

# Oznitelikler ve siniflar ayrildi.

x = ttt.drop(columns='class')
y = ttt['class']

#null deger kontrolu yapildi.

ttt.isnull().any() # Veri setinde hic bos deger yok.

#Veri On isleme kapsaminda x ve y' ye gerekli donusumu yapalim.
from sklearn.preprocessing import LabelEncoder
le_encoder = LabelEncoder()

x = pd.get_dummies(x) # golge degiskene donusturuldu.

y = le_encoder.fit_transform(y) #labelencoding uygulandi.

# K-Means sinifini yukleyelim.
from sklearn.cluster import KMeans
# KMeans algoritmas nda daha  nce  de anlatildigi gibi oncelikle "optimum K degeri" belirlenmesi
# Bunun icin Dirsek Y ntemi (The Elbow Method) kullancagiz.

# Bos bir liste olusturuldu. Her bir K degeri icin
# bulunan WCSS degerleri bu listeye atildi.

WCSS =[]
for K in range(1,11):
    KM = KMeans(n_clusters = K, init= 'k-means++', max_iter =300, n_init = 10)
    KM.fit(x) # Daha once belirledigimiz x parametresini algoritmaya veriyoruz.
    WCSS.append(KM.inertia_)

# KMeans algoritmasinda en buyuk problemlerden biri rastgele
# baslangic noktasi sorunudur.
# k-means++ parametresi bizi bu durumdan kurtaracak sekilde
# baslangic noktalari seciyor.

# Dirsek metodunun grafigini gorelim.

plt.plot(range(1,11),WCSS)
plt.title ('Tic Tac Toe veri seti icin Dirsek Y ntemi')
plt.xlabel ('Kume Sayisi')
plt.ylabel ('WCSS')
plt.show()

np.diff(WCSS)

model_predicted = ['y_KM','y_KNN','y_DT','y_NB','y_LR','y_SVM']

# O halde K=2 icin K-Means uygulayalim.

KM = KMeans(n_clusters = 2, init= 'k-means++', max_iter =300, n_init = 10)
model_predicted[0]= KM.fit_predict(x)

# Cross Validation icin gerekli kutuphaneleri yuklendi.
```

```
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
from sklearn.metrics import cohen_kappa_score,confusion_matrix
from sklearn.model_selection import KFold,cross_val_score,cross_val_predict

# 10-fold cross validation yapacagimiz icin KFold fonksiyonuna gerekli
# parametreler verildi.

KFCV = KFold(n_splits = 10, random_state = 42)

# K En Yakin Komsu sinifini Olusturalim.
# Fakat Once optimum K degerini belirleyelim.

from sklearn.neighbors import KNeighborsClassifier

values =[]

for K in range(1,20):  # bu araliktaki her bir K degeri icin KNN calisacak
ve en yuksek dogrulugu olani alacagiz.
    KNN = KNeighborsClassifier(n_neighbors=K,metric = 'euclidean')
    best_accuracies = cross_val_score(estimator=KNN,X=x,y=y,cv=KFCV,scoring='accuracy').mean()
    values.append(best_accuracies)

plt.plot(range(1,20),values)
plt.ylabel('Dogruluk')
plt.xlabel('K Sayisi')

print('En iyi dogruluk degerine sahip K degeri:', values.index(max(values)),','
    ,'Dogruluk:', max(values))

# Bulunan optimum K degeri ile model calistirildi.

KNN = KNeighborsClassifier(n_neighbors=values.index(max(values)),metric='euclidean')

#Karar Agaci sinifini olusturalim. Dallanma kriteri olarak Entropi kullanalim.

from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier(criterion = 'entropy')

#Naive Bayes sinifini olusturalim.

from sklearn.naive_bayes import GaussianNB
NB = GaussianNB()

#Logistik Regresyon sinifini olusturalim.

from sklearn.linear_model import LogisticRegression
LogR = LogisticRegression(solver = 'liblinear') # Veri setimiz buyuk bir veri seti
# olmadigindan solver icin 'liblinear' iyi bir secenek.
# Daha buyuk veri setlerinde 'sag' veya 'saga' secilebilir.
#Ayrica veri setimiz iki sinifli oldugundan multi_class ozelligini kullanmadik.

#SVM icin svm sinifi olusturuldu.

from sklearn import svm
L_SVM = svm.LinearSVC()  # Lineer SVM' i kullanalim.

# Kullanacagimiz metrikler icin bir dizi olusturuldu.
scoring = ['accuracy' ,'precision','recall','f1']


model_estimator_list=[KNN,DT,NB,LogR,L_SVM]
model_name_list=['K-Nearest Neighbors','Decision Tree','Naive Bayes', 'Logistic Regression','SVM']


# Modellerin hata matrislerine bakalim.


print('K-Means icin hata matrisi: \n' ,confusion_matrix(y,model_predicted[0]))

for i in range(1,6):
    model_predicted[i]=cross_val_predict(model_estimator_list[i-1],x,y,cv=KFCV)
    print(model_name_list[i-1],'icin hata matrisi: \n',
          confusion_matrix(y,model_predicted[i]))
```

```
# Siniflandirma sonrasinda elde edilen metriklere bakalim.

# KMeans' in ayri degerlendirilmesinin sebebi uzerinde cross validation
# uygulanmamis olmasi.

print('K-Means modeli icin accuracy', accuracy_score(y,model_predicted[0]))
print('K-Means modeli icin precision', precision_score(y,model_predicted[0]))
print('K-Means modeli icin recall', recall_score(y,model_predicted[0]))
print('K-Means modeli icin f1', f1_score(y,model_predicted[0]))


for i in range(0,5):
  for j in range(0,4):
     metrics_scores=cross_val_score(estimator=model_estimator_list[i],X=x,y=y,
                                          cv=KFCV,scoring=scoring[j])
     print(model_name_list[i],'modeli icin', scoring[j],metrics_scores.mean()
     print(model_name_list[i],'toplam', scoring[j],metrics_scores.mean())

# Her bir modelin Cohen Kappa Skorlarini hesaplayalim.

print('K-Means modeli icin kappa',cohen_kappa_score(y,model_predicted[0]))
for i in range(0,5):
   print(model_name_list[i],'icin kappa',cohen_kappa_score(y,model_predicted[i]))
```

ÖZGEÇMİŞ

Ad Soyad: Kerem Kabil

Doğum Tarihi ve Yeri: 26.02.1992, İstanbul

E-Posta: kkeremkabil@gmail.com

ÖĞRENİM DURUMU:

- **Lisans:** 2016, Yıldız Teknik Üniversitesi, Fen Edebiyat Fakültesi, Matematik
- **Y. Lisans:** 2019, İstanbul Teknik Üniversitesi, Matematik Mühendisliği Anabilim Dalı, Matematik Mühendisliği.

MESLEKİ DENEYİMLER VE ÖDÜLLER:

- 2016 yılında Yıldız Teknik Üniversitesi' nde lisans eğitimini tamamladı.
- 2019 yılında İstanbul Teknik Üniversitesi' nde yüksek lisans eğitimini tamamladı.