

# Kırmızı Şarap Kalitesinin Makine Öğrenmesi Kullanılarak Tahmin Edilmesi

## Predicting Red Wine Quality Using Machine Learning

Mehtap ÖKLÜ  
Bilgisayar Mühendisliği  
17110131052

Kahramanmaraş Sütçü İmam Üniversitesi  
Kahramanmaraş, Türkiye  
mehtap\_oklu\_06@hotmail.com

Banu KÖSE  
Bilgisayar Mühendisliği  
18110131011

Kahramanmaraş Sütçü İmam Üniversitesi  
Kahramanmaraş, Türkiye  
banukose1561@gmail.com

**Özetçe** —Şarap, çok eski tarihlerden beri tüketilen, batı toplumlarında mutfakla özdeşleşmiş olan alkollü bir içecektir. Farklı meyvelerle üretiliyor olsa da, şarap denince akla ilk gelen meyve üzümdür. Şarap, çok geniş bir skalaya sahiptir. Bu yüzden her şarabın kalite oranı farklıdır. Üzümlle üretilen şaraplardan biri de kırmızı şaraptır. Kırmızı şaraba ait bazı öznel özelliklerin (alkol oranı, pH, klorür vb.) değerleri incelenerek şarabın kalitesi tahmin edilebilir. Bu değerlerin incelenmesi için de belirlediğimiz bazı yapay zeka algoritmalarını (KNN, SVC, Lojistik Regresyon, Naive Bayes, Karar Ağacı, Bagging ve Rastgele Orman Ağacı) kullandık. Kullandığımız algoritmalar genellikle sınıflandırma algoritmalarıdır. Bunun sebebi, şarabı kaliteli veya kalitesiz (0-1) olarak etiketlemek istememizdir.

**Anahtar Kelimeler**—Kırmızı Şarap, Kalite, KNN, SVC, Lojistik Regresyon, Naive Bayes, Karar Ağacı, Bagging ve Rastgele Orman Ağacı

**Abstract**—Wine is an alcoholic beverage that has been consumed since time immemorial, identified with cuisine in western societies. Although it can be produced with different fruits, the first fruit that comes to mind when it comes to wine is grapes. Wine has a very Large scale. That is why the quality ratio of each wine is different. One of the wines produced with grapes is red wine. It should be noted that some attributes of red wine (alcohol content, pH, chloride, etc.) the quality of the wine can be estimated by examining its values. In order to examine these values, we used some artificial intelligence algorithms (KNN, SVC, Logistic Regression, Naive Bayes, Decision Tree, Bagging, and Random Forest Tree) that we also determined. The algorithms we use are usually classification algorithms. This is because we want to label the wine as good quality or poor quality (0-1).

**Keywords**—Red Wine, Quality, KNN, SVC, Logistic Regression, Naive Bayes, Decision Tree, Bagging, and Random Forest Tree

### I. GİRİŞ

Şarap, diğer adıyla mey. Parçalanmış veya parçalanmamış üzümün fermente edilmesiyle [1] üretilen, tescillenmiş veya tescillenmemiş, %9-15 oranda alkole sahip [2] bir içecektir. Tescillenmiş şaraplar, üretildiği coğrafi bölgenin bir işaretini taşır [3].

Şarap, bilinmeyen tarihlerden beri üretilen ve tüketilen bir içecektir. Elimizdeki bilgilere göre şu an şarabın bilinen doğum yeri Gürcistan'dır, tarihi ise MÖ 6000'lere kadar dayanmaktadır [2]. Şarap, birden fazla çeşide sahiptir. Bunlar; kırmızı, beyaz, rose, blush, likör şarabı vb. şeklindedir [1]. Bu makalede kırmızı şarabı inceleyeceğiz.

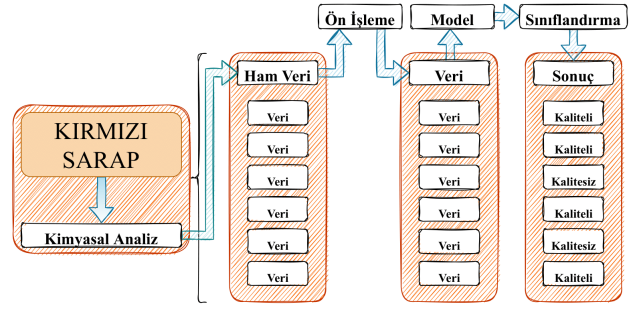
Şarap üretimi çok fazla emek isteyen, yorucu bir iştir. Gerek ilkel yöntemlerle, gerek de modern teknolojiyle şarap üretmek mümkündür. Şarap üretiminde yer alan bu teknolojinin ilerlemesiyle birlikte, şarap kalitesiyle alakalı kriterler de daha detaylı hale gelmiştir [1]. Bu kriterler ve parametreler (uçucu asitlik, artık şeker miktarı, alkol oranı, pH, klorür vb.) daha net bir şekilde incelenebildiği için şarabın kalitesiyle ilgili ölçümler de daha rahat yapılabilir. Bu makalede de yapay zeka kullanarak kırmızı şarap kalitesini ölçmeye çalışacağız.

Kırmızı şarap üzerinde kimyasal ölçümler yapılarak bir takım veriler elde edildi. Bu veriler de Kaggle [4] üzerinde açık kaynak olarak paylaşıldı. Biz de bu veri setini edindik, üzerinde ön işleme yaptık ve üzerine makine öğrenmesi algoritmalarını kurduk. Bu makalemizde de makine öğrenmesi kullanarak kırmızı şarap kalitesini tahmin etmeye çalışacağız.

Bu projede “Red Wine Quality” isimli veri setin [4] Kaggle üzerinden temin ettik. Verinin %30’unu test için, %70’ini de eğitim için kullandık. Veri setinde 11 adet öznitelik, 1600 adet kayıt var. Bu öznitelikler:

- **fixed acidity (sabit asitlik):** Şarapla ilgili sabit, uçucu olmayan asitler. Kolayca buharlaşmazlar [5].
- **volatile acidity (uçucu asitlik):** Şarapta yüksek seviyelerde bulunduğu sirke tadını ortaya çıkmasına neden olabilir [5].
- **citric acid (sitrik asit):** Şarapta az miktarda bulunduğu daha taze ve fresh bir tat verir [5].
- **residual sugar (artık şeker):** Fermantasyon işleminden sonra arta kalan şeker miktarı, 1 g/L’den az olan şaraplar nadir bulunur. 45 g/L’den fazla şaraplar sweet(tatlı) olarak kabul edilir [5].
- **chlorides (klorür):** Şaraptaki tuz miktarıdır [5].
- **free sulfur dioxide (serbest kükürt dioksit):** Serbest form halinde bulunan kükürt dioksit, çözünmüş bir gaz ve bisülfid iyonu arasında dengede kalır.. Şarap oksidasyonunu önleme yanında büyümeyi (mikrobiyal) sağlar [5].
- **total sulfur dioxide (toplam kükürt dioksit):** Düşük seviyelerde, şarapta anlaşılması zordur fakat 50PPM’yi geçtiği zaman şarabın kokusunda ve tadında belirgin hale gelir [5].
- **density (yoğunluk):** Şarabın yoğunluğu 1.08 - 1.09 civarındadır. Bu durum da şarabın suya göre %8-9 daha yoğun olduğu anlamına gelir [6].
- **pH (asidik-bazik):** pH cetvelinde 0-14 arasında değerde şarabın asitlik ve baziklik değerini verir. Şaraplar pH cetvelinde 3-4 arasında yer alır [5].
- **sulphates (sülfür):** Antimikrobiyal ve antioksidan görevi gören kükürt dioksit gazının (SO2) seviyelerine katkıda bulunabilen bir katkı maddesidir [5].
- **alcohol:** Şarabın yüzde kaç alkol içerdiğini gösterir. Kırmızı şarapta alkol oranı %11-14 arasındadır [5].

Bu veri setinde kalite sütunu yani sonuç özniteliğinde 0-1 değerleri yer alıyor. Öznitelikler doğrultusunda şarabın yüksek veya düşük kaliteli olduğunu belirtiyor. Elimizdeki bu veri setinde en verimli şekilde etiketleme yapabilmek için; KNN, SVC, Lojistik Regresyon, Naive Bayes, Karar Ağacı, Bagging ve Rastgele Orman Ağacı Algoritmalarını kullanmaya karar verdik. Projemizi Python dilinde, Anaconda-Spyder editöründe yazdık.



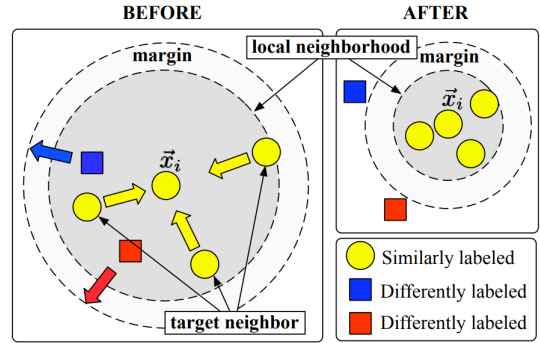
Şekil 1: Kırmızı Şarap Kalite Kontrol Süreci

Projemizin akış diyagramı Şekil 1’deki gibidir. Makalenin ikinci bölümünde, bahsettiğimiz algoritmalar hakkında bilgiler sunulmaktadır. Daha sonraki bölümde ise sonuçlar yer almaktadır.

## II. KULLANILAN ALGORİTMALAR

### A. K-Nearest Neighbor

Gözetimli (denetimli) öğrenme metotlarından olup basit sınıflandırma işlemlerinde kullanılır [8] [9].Sınıflandırma çalışması yaparken elimizdeki veriler hakkında kısıtlı ön bilgiye sahip olduğumuzda tercih etmemiz gereken ilk algoritmalarından biridir [8].



Şekil 2: Örnek KNN Şeması [10]

Bu algoritmanın performansını etkileyen parametreler; uzaklık ölçütü, komşu sayısı(k) ve ağırlıklandırma yöntemidir [7]. Uzaklık ölçütü olarak Manhattan uzaklığı kullandık:  $n$  boyutlu düzlemdeki iki konum arasındaki farkların, mutlak değerlerinin toplamıdır.  $X$ - $Y$  konumları arasındaki Manhattan uzaklığı:  $P=(x_1, x_2, \dots, x_n)$  ve  $Q=(y_1, y_2, \dots, y_n)$  olmak üzere, Eşitlik 1’e göre hesaplanır [7].

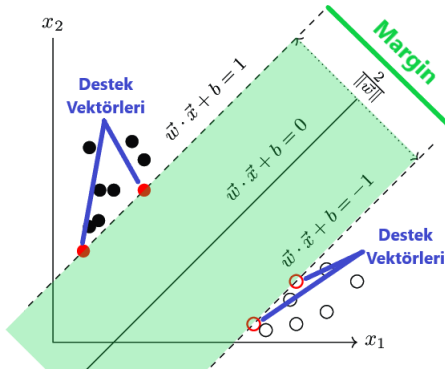
$$\sum_{i=1}^n |x_i - y_i| \quad (1)$$

KNN algoritmasının aşamaları [9]:

- K değeri belirlenir
- Diğer konumlara olan uzaklık, Manhattan yöntemiyle hesaplanır
- Uzaklıklar sıralanarak en yakın komşular bulunur
- En yakın K adet komşunun kategorileri toplanır
- Toplam sonucunda ağırlıkta olan kategori seçilerek etiketleme yapılır

### B. Support Vector Machines (SVM)

Türkçe adıyla Destek Vektör Makineleri (DVM), istatistiksel öğrenme teorisi geliştiricisi Vapnik tarafından geliştirilmiştir. Sınıflandırma, örüntü tanıma problemleri için kullanılır. Destek Vektör Makineleri, alanındaki birçok tekniğe göre daha yüksek başarı oranına sahiptir. Uygulama sırasında çekirdek fonksiyon seçimi ve parametre optimizasyonu çok önemlidir [11].



Şekil 3: Örnek SVM Şeması [12]

Şekil 3 üzerinde siyah ve beyaz olmak üzere iki sınıf mevcut. Gelecek verilerin sınıflandırılabilmesi için öncelikle sınıfları birbirinden ayıran bir doğru çizilir. Bu doğrunun -1 ve +1 değerleri arasında kalan yeşil kısım Margin bölgesidir. Margin ne kadar geniş olursa, sınıflar o kadar iyi ayrılıyor demektir. Bu işlemin formülü Eşitlik 2'deki gibidir.

$$\hat{y} = \begin{cases} 0, w^T \cdot x + b < 0, \\ 1, w^T \cdot x + b \geq 0 \end{cases} \quad (2)$$

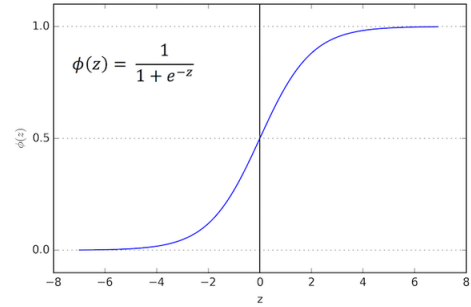
Elde ettiğimiz sonuç 0'dan küçük çıkarsa beyaz noktaların bulunduğu kısma yakın olacaktır. Sonuç 0'a eşit veya 0'dan büyük çıkarsa siyah noktaların bulunduğu kısma daha yakın olacaktır [12].

### C. Logistic Regression

Lojistik Regresyon, bize sonucu 0 veya 1 şeklinde üreten, değişkenlerin modellenmesinde kullanılan algoritmadır. Örneğin kırmızı şarap kalitesini ele aldığımız zaman; elimizdeki veri setinde 0 kalitesiz, 1 kaliteli durumu temsil eder. Kısacası Lojistik Regresyon yöntemiyle kullanılan örneklemelerin hangi sınıfa ait olduklarını belirleyebiliriz [13].

$$f(x) = \frac{1}{1 + e^{-z}} \quad (3)$$

Lojistik regresyon,  $(-\infty, +\infty)$  aralığındaki değerleri girdi olarak kabul eder [13].



Şekil 4: Örnek bir Lojistik Regresyon Şeması [13]

### D. Naive Bayes

Navie Bayes algoritması sınıflandırıcı teorem olan Bayes'e dayanmaktadır [15]. Adını İngiliz matematikçi olan Thomas Bayes'ten almıştır [14]. Navie Bayes, elimizdeki örneklerin hangi sınıfa ait olduğunu, işlemler sonucu elde ettiği oranla tahmin eder. Navie Bayes'in iki önemli kabulü vardır [15]. Bunlar:

- Elimizde bulunan niteliklerin hepsi aynı derecede önemlidir.
- Elimizde bulunan nitelikler birbirinden bağımsızdır.

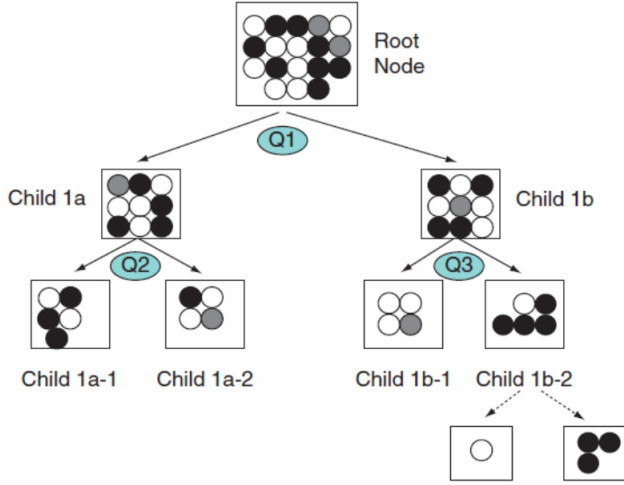
Koşullu olasılıklar arasında bağlantı kuran Bayes Teoremi, x özneliliğinin gerçekleşmiş olması durumunda C özneliliğinin gerçekleşmesi veya C özneliliğinin gerçekleşmesi durumunda x özneliliğinin gerçekleşmesi durumu bize verir [15]. Bayes kuralının algoritması Eşitlik 4'deki gibidir. Eşitlikte yer alan değerler [14]:

- $p(x|C_j)$ : j sınıfından bir örneğin x olma olasılığı
- $P(C_j)$ : j sınıfının ilk olasılığı
- $p(x)$ : Herhangi bir örneğin x olma olasılığı
- $P(C_j|x)$ : x olan bir örneğin j sınıfından olma olasılığı (son olasılık)

$$P(C_j|x) = \frac{p(x|C_j)P(C_j)}{p(x)} = \frac{p(x|C_j)P(C_j)}{\sum_k p(x|C_k)P(C_k)} \quad (4)$$

### E. Decision Tree

Karar ağaçları, en yaygın kullanılan gözetimli(denetimli) öğrenme algoritmalarından birisidir. Adından da anlaşılacağı üzere ağaç tabanlı öğrenmeye dayalıdır. Tüm problemlere uyarlanabilir [16]. Karar ağacı, kök düğüm değişkeninden başlayarak, ağaç dalı hiyerarşisi gibi dallanır [17].



Şekil 5: Örnek bir Karar Ağacı Şeması [17]

Entropi, verilerimizle alakalı belirsizliğin bir ölçüsüdür; dolayısıyla entropinin düşük olması tercih edilir. Sezgisel olarak verilerin tamamı aynı etikete sahipse, veri setinin entropisinin düşük olduğunu söyleyebiliriz [16].

$$H = - \sum p(x) \log p(x) \quad (5)$$

Burada;  $p(x)$  belirli bir sınıfa ait grubun yüzde oranını,  $H$  ise entropiyi belirtmektedir. Karar ağacı, entropiyi en aza indirecek şekilde bölünme yapmalıdır. En iyi bölümlemeyi belirlemek için kullanılan bilgi kazancı, Eşitlik 6'daki gibi hesaplanır [16].

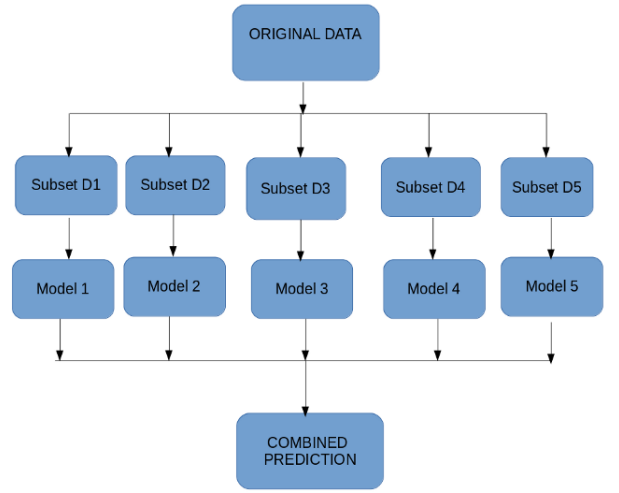
$$Gain(S, D) = H(S) - \sum_{V \in D} \frac{|V|}{|S|} H(V) \quad (6)$$

Burada;  $S$  orijinal veri kümesini,  $D$  ise kümenin bölünmüş bir parçasını temsil eder. Her  $V$  değeri,  $S$ 'nin alt kümesidir [16].

### F. Bagging

Türkçe adıyla torbalama yöntemi olarak adlandırılan Bagging algoritması 1996 yılında Breiman tarafından geliştirilmiştir. Bagging (torbalama) algoritmasının çalışma biçimi Şekil 6'da belirtilmiştir. Şekil 6'daki aşamaların açıklamaları [18]:

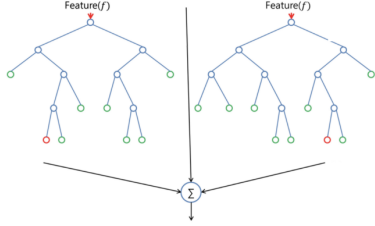
- **Bootstrap sampling:** Orijinal veri kümesi, alt kümelere bölünür.
- **Model training:** Bu alt kümelerin her birinde ayrı ayrı temel (zayıf) model oluşturulur.
- **Model forecasting:** Modeller birbirinden bağımsız ancak paralel olarak çalışır.
- **Result aggregating:** Tüm modellerin tahmin sonuçları birleştirilerek nihai tahmin belirlenir.



Şekil 6: Örnek bir Torbalama Şeması [18]

### G. Random Forest Tree

Türkçe adıyla Rastgele Orman Ağacı Algoritması, denetimli bir sınıflandırma algoritmasıdır. Adından da anlaşılacağı gibi, algoritmamız rastgele bir orman yaratıyor. Bu ormandaki ağaç sayısı ve elde edilebilecek sonuçlar arasında doğrusal bir ilişki vardır. Ağaç sayısı arttıkça elde edilecek olan sonucun kesinliği de artar. Rastgele Orman Ağacı'nın Karar Ağacı'ndan farkı; kök bulma ve düğümleri bölme işlemlerinin çalışıyor olmasıdır [19]. Rastgele Orman Ağacı'nın şeması Şekil 7'de verilmiştir.



Şekil 7: Örnek bir Rastgele Orman Ağacı Şeması [20]

Rastgele Orman Ağacı, kullanıcıdan iki parametre alır:  $m$  parametresi en iyi bölünmeyi belirlemek için her düğümde kullanılan değişkenlerin sayısı,  $N$  geliştirilecek ağaç sayısı. Öncelikle eğitim verilerinin  $2/3$ 'ü kullanılarak önyükleme örnekleri oluşturulur. Geri kalan  $1/3$ 'lük kısım (OOB: out of bag) ise hataları test etmek için kullanılır [21].

Her bir düğümde  $m$  değişkenleri, tüm değişkenlerin arasından rastgele seçilerek en iyi dal belirlenir.  $M$  adet değişkenin kareköküne eşit olarak alınan  $m$  değişken, genellikle optimuma en yakın sonucu verir [21].

Sınıfların homojenliği *Gini* indexi hesaplanarak ölçülür. *Gini* indexi ne kadar düşükse, sınıf da o kadar homojendir. Bir düğümün alt *Gini* indexi üst *Gini* indexinden daha az olduğu durumlarda incelenen dal başarılı sayılır [21]. *Gini* indexinin formülü Eşitlik 7'de verilmiştir. Formül değişkenlerinin temsil ettiği veriler de aşağıdaki gibidir;

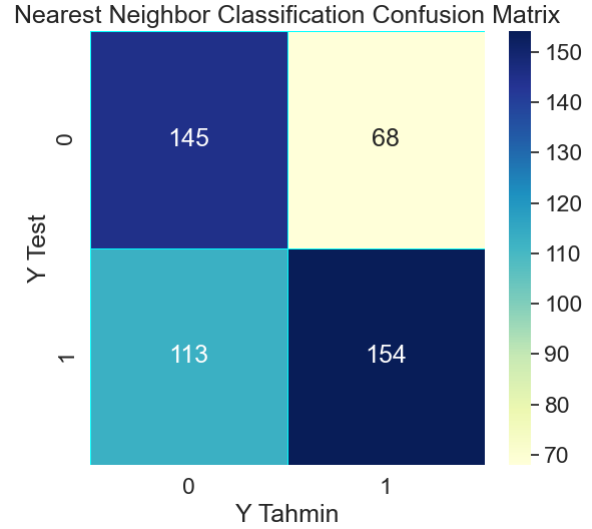
- **T:** Tüm veri seti
- **pj:** Veri setindeki her bir verinin kendinden küçük ve kendünden büyük eleman sayılarına bölümü
- **n:** Seçilen verimiz

$$Gini(T) = 1 - \sum_{j=1}^n (p_j)^2 \quad (7)$$

### III. DENEYSEL SONUÇLAR VE TARTIŞMA

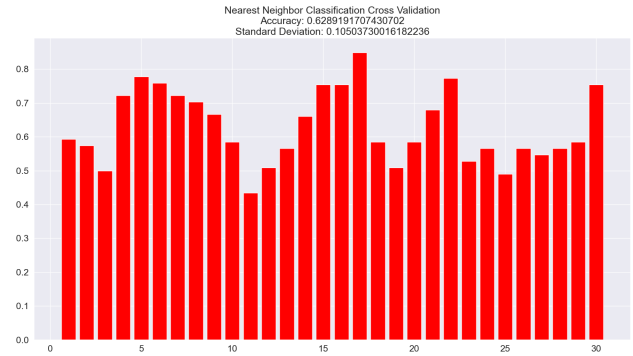
#### A. K-Nearest Neighbor

Şekil 8'de **K-Nearest Neighbor** algoritmamızın 8'de hata matrisinin görselini görüyoruz. Hata matrisindeki değerleri kullanarak: **Doğruluk değeri:**  $(145 + 154) / (145 + 68 + 113 + 154) = 0.62$  **Hata oranı:**  $1 - DoğrulukDeğeri(0.62) = 0.38$



Şekil 8: K-Nearest Neighbor Hata Matrisi

**K-Nearest Neighbor** algoritmasını Cross-Validation'a soktuk ve 30 kere rastgele verilerle çalışmasını sağladık. Bu değerlerin grafiği Şekil 9'daki gibidir. Şekildeki değerlerin ortalaması 0.6289, standart deviasyonu ise 0.1050'dir.

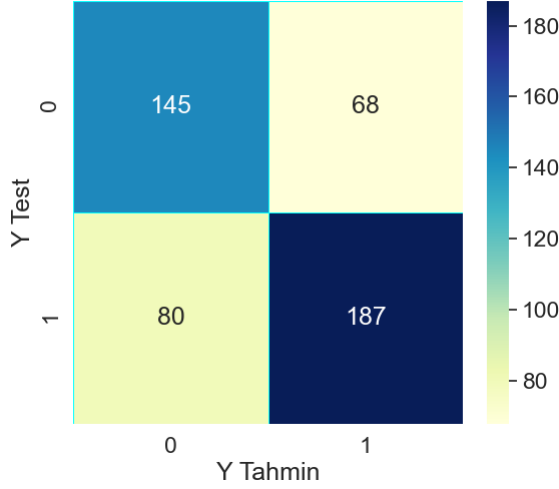


Şekil 9: K-Nearest Neighbor Cross-Validation Değerleri

## B. Support Vector Machines (SVM)

Şekil 10'da Support Vector Machines algoritmamızın hata matrisinin görselini görüyoruz. Hata matrisindeki değerleri kullanarak: **Doğruluk değeri:**  $(145 + 187) / (145 + 68 + 80 + 187) = 0.69$   
**Hata oranı:**  $1 - DoğrulukDeğeri(0.69) = 0.31$

C-Support Vector Classification Confusion Matrix

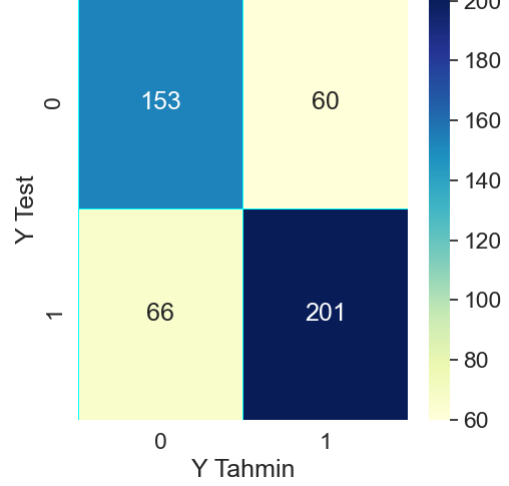


Şekil 10: Support Vector Machines Hata Matrisi

## C. Logistic Regression

Şekil 12'de Logistic Regression algoritmamızın hata matrisinin görselini görüyoruz. Hata matrisindeki değerleri kullanarak: **Doğruluk değeri:**  $(153 + 201) / (153 + 60 + 66 + 201) = 0.73$   
**Hata oranı:**  $1 - DoğrulukDeğeri(0.73) = 0.27$

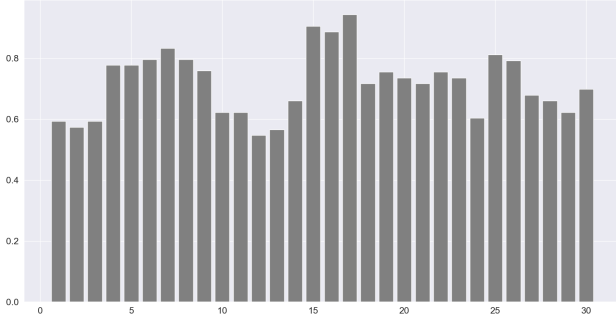
Logistic Regression Confusion Matrix



Şekil 12: Logistic Regression Hata Matrisi

Support Vector Machines algoritmasını Cross-Validation'a soktuk ve 30 kere rastgele verilerle çalışmasını sağladık. Bu değerlerin grafiği Şekil 11'deki gibidir. Şekildeki değerlerin ortalaması 0.7179, standart deviasyonu ise 0.1032'dir.

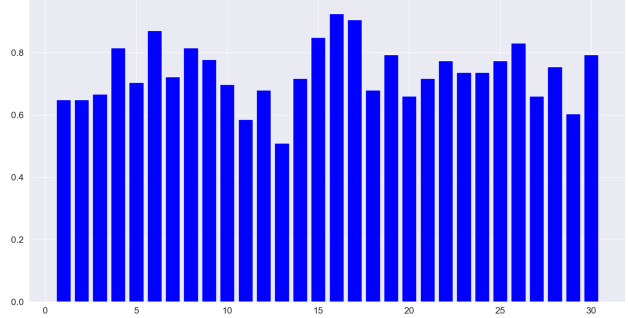
C-Support Vector Classification Cross Validation  
 Accuracy: 0.7179245283018869  
 Standard Deviation: 0.1032503981768412



Şekil 11: Support Vector Machines Cross-Validation Değerleri

Logistic Regression algoritmasını Cross-Validation'a soktuk ve 30 kere rastgele verilerle çalışmasını sağladık. Bu değerlerin grafiği Şekil 13'deki gibidir. Şekildeki değerlerin ortalaması 0.7348, standart deviasyonu ise 0.0932'dir.

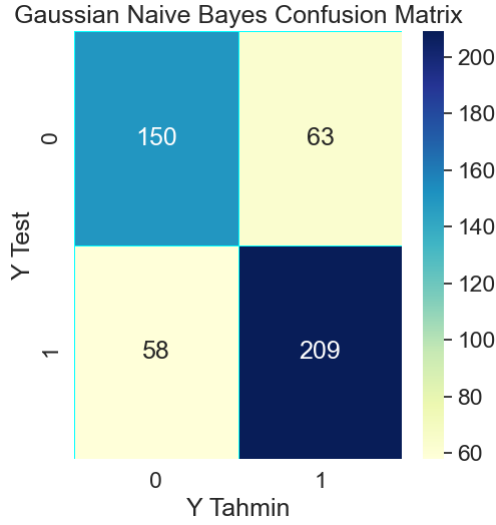
Logistic Regression Cross Validation  
 Accuracy: 0.7348008385744237  
 Standard Deviation: 0.0932508280489236



Şekil 13: Logistic Regression Cross-Validation Değerleri

#### D. Naive Bayes

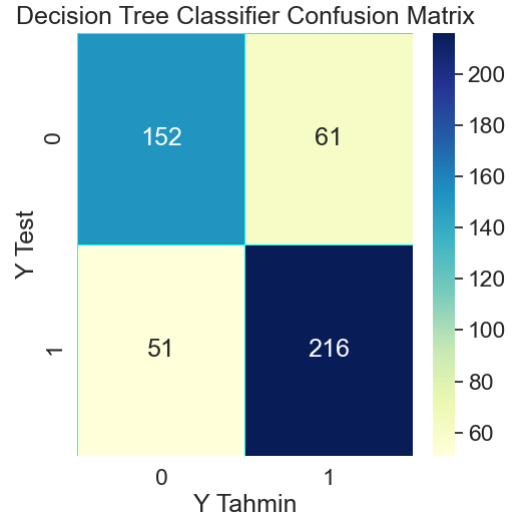
Şekil 14’de **Naive Bayes** algoritmamızın hata matrisinin görselini görüyoruz. Hata matrisindeki değerleri kullanarak: **Doğruluk değeri:**  $(150 + 209) / (150 + 63 + 58 + 209) = 0.74$   
**Hata oranı:**  $1 - DoğrulukDeğeri(0.73) = 0.26$



Şekil 14: Naive Bayes Hata Matrisi

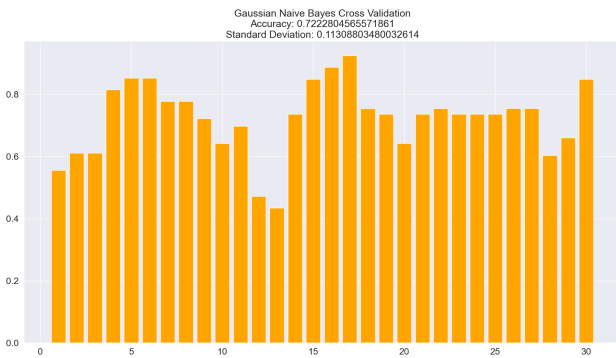
#### E. Decision Tree

Şekil 16’da **Decision Tree** algoritmamızın hata matrisinin görselini görüyoruz. Hata matrisindeki değerleri kullanarak: **Doğruluk değeri:**  $(152 + 216) / (152 + 61 + 51 + 216) = 0.76$   
**Hata oranı:**  $1 - DoğrulukDeğeri(0.76) = 0.24$



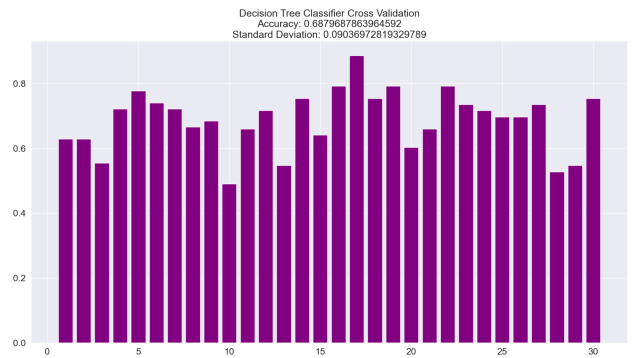
Şekil 16: Decision Tree Hata Matrisi

**Naive Bayes** algoritmasını Cross-Validation’a soktuk ve 30 kere rastgele verilerle çalışmasını sağladık. Bu değerlerin grafiği Şekil 15’deki gibidir. Şekildeki değerlerin ortalaması 0.7222, standart deviasyonu ise 0.1130’dur.



Şekil 15: Naive Bayes Cross-Validation Değerleri

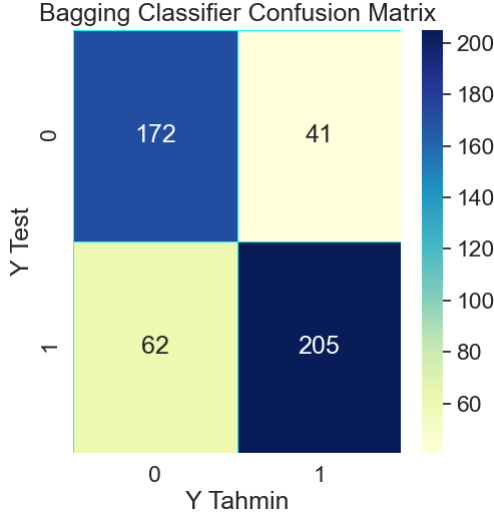
**Decision Tree** algoritmasını Cross-Validation’a soktuk ve 30 kere rastgele verilerle çalışmasını sağladık. Bu değerlerin grafiği Şekil 17’deki gibidir. Şekildeki değerlerin ortalaması 0.6879, standart deviasyonu ise 0.0903’dur.



Şekil 17: Decision Tree Cross-Validation Değerleri

## F. Bagging

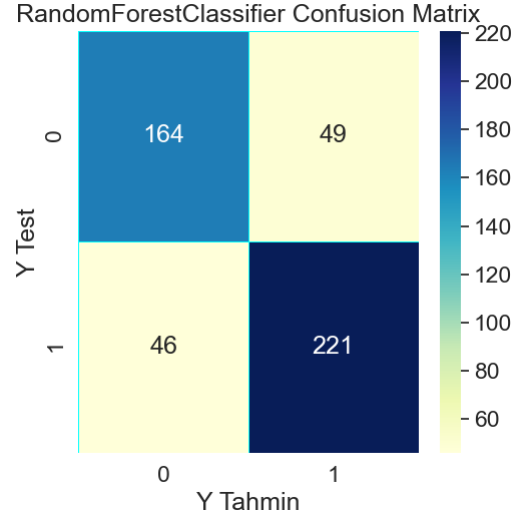
Şekil 18'de **Bagging** algoritmamızın hata matrisinin görselini görüyoruz. Hata matrisindeki değerleri kullanarak: **Doğruluk değeri:**  $(172 + 205)/(172 + 41 + 62 + 205) = 0.78$   
**Hata oranı:**  $1 - DoğrulukDeğeri(0.78) = 0.22$



Şekil 18: Bagging Hata Matrisi

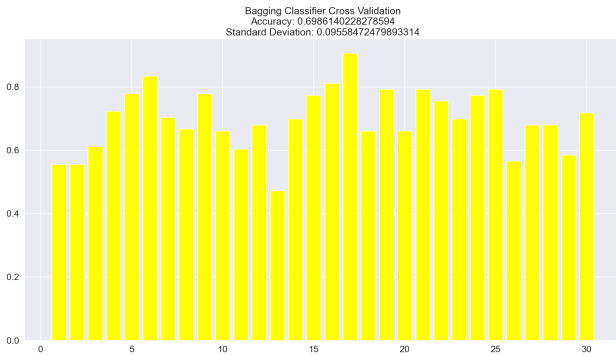
## G. Random Forest Tree

Şekil 20'de **Random Forest Tree** algoritmamızın hata matrisinin görselini görüyoruz. Hata matrisindeki değerleri kullanarak: **Doğruluk değeri:**  $(164 + 221)/(164 + 49 + 46 + 221) = 0.80$   
**Hata oranı:**  $1 - DoğrulukDeğeri(0.80) = 0.20$



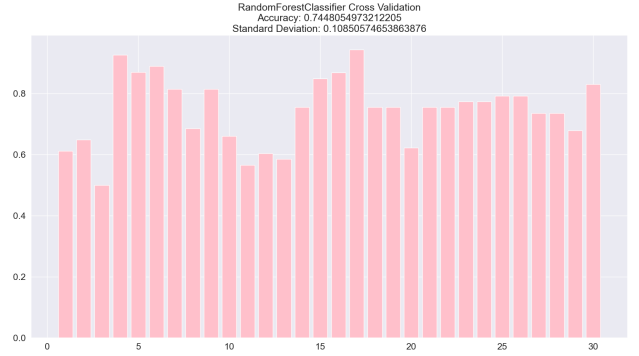
Şekil 20: Random Forest Tree Hata Matrisi

**Bagging** algoritmasını Cross-Validation'a soktuk ve 30 kere rastgele verilerle çalışmasını sağladık. Bu değerlerin grafiği Şekil 19'daki gibidir. Şekildeki değerlerin ortalaması 0.6986, standart deviasyonu ise 0.0955'dur.



Şekil 19: Bagging Cross-Validation Değerleri

**Random Forest Tree** algoritmasını Cross-Validation'a soktuk ve 30 kere rastgele verilerle çalışmasını sağladık. Bu değerlerin grafiği Şekil 21'deki gibidir. Şekildeki değerlerin ortalaması 0.7448, standart deviasyonu ise 0.1085'dur.



Şekil 21: Random Forest Tree Cross-Validation Değerleri



## H. Tüm Algoritmaların Ait Sonuçlar & ROC Eğrisi Grafiği

Belirlediğimiz algoritmaları çalıştırdık ve sonuçlarını aldık. Bu sonuçları görsel olarak Şekil 22’de belirttik. Sonuçlar liste olarak:

### Nearest Neighbor Classification = Kırmızı

- Doğruluk: 0.62
- MSE: 0.37

### C-Support Vector Classification = Gri

- Doğruluk: 0.69
- MSE: 0.30

### Logistic Regression = Mavi

- Doğruluk: 0.73
- MSE: 0.26

### Gaussian Naive Bayes = Turuncu

- Doğruluk: 0.74
- MSE: 0.25

### Decision Tree Classifier = Mor

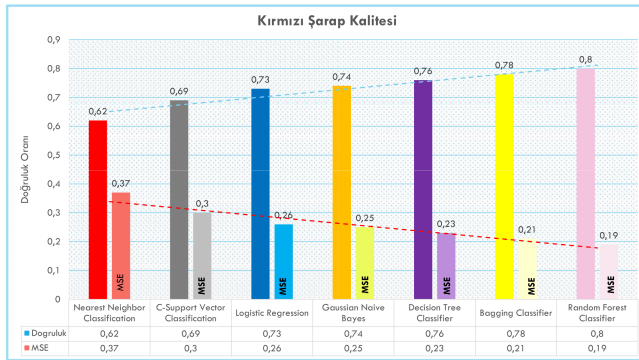
- Doğruluk: 0.76
- MSE: 0.23

### Bagging Classifier = Sarı

- Doğruluk: 0.78
- MSE: 0.21

### RandomForestClassifier = Pembe

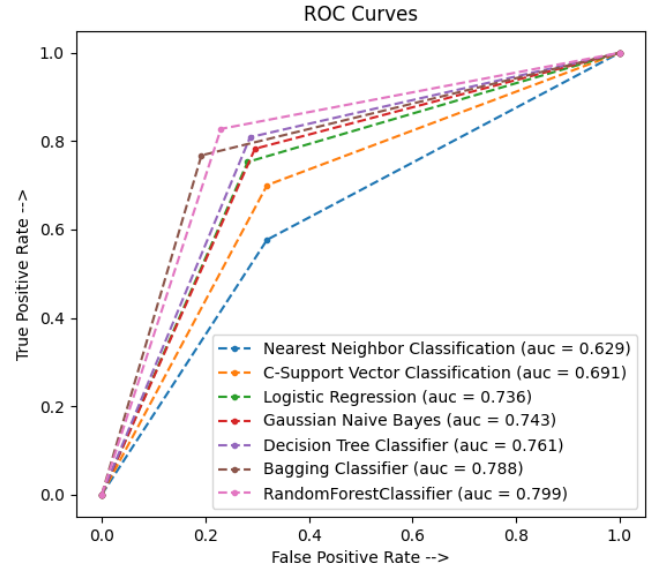
- Doğruluk: 0.80
- MSE: 0.19



Şekil 22: Tüm Algoritmaların Doğruluk & MSE Oranları

Makine öğrenmesinde performans ölçümünde ROC eğrisinden yararlanılır. Bu eğrilerin AUC(doğruluk) değerleri de mevcuttur ancak algoritmanın tahmin doğruluk skoruyla karıştırılmamalıdır [22]. Elimizdeki algoritmaların ROC eğrilerini Şekil 23’de belirttik. Eğrilerin doğruluk oranları da listedeki gibidir:

- Nearest Neighbor Classification: 0.62
- C-Support Vector Classification: 0.69
- Logistic Regression: 0.73
- Gaussian Naive Bayes: 0.74
- Decision Tree Classifier: 0.76
- Bagging Classifier: 0.78
- RandomForestClassifier: 0.79



Şekil 23: Tüm Algoritmaların ROC Eğrileri ve AUC Değerleri

Kullandığımız algoritmalarından en başarılı olan Random Forest Tree algoritması oldu. %80 lik doğruluk oranıyla diğer algoritmaları geride bıraktı. Ortalama hata karesi değeri diğer algoritmalarla göre daha düşük (0.19) çıktı. En düşük başarıya sahip olan algoritmamız ise %62’lik başarı oranıyla K-Nearest Neighbor algoritması oldu. Ortalama hata karesi değerini de Random Forest Tree’ninkine göre yüksek kalan 0.37 olarak elde ettik. Algoritmaların bu değerlerini Şekil 22’de belirttik. Diğer algoritmalar ise birbirine oldukça yakın değerler verdi. Bagging algoritması ise %78 doğruluk oranı verirken 0.21 MSE değeriyle, en verimli ikinci algoritma oldu. Kullandığımız en verimli üçüncü algoritma ise %76 doğruluk değeriyle Decision Tree algoritması oldu. MSE değeri olarak da 0.23 verdi. Decision Tree’nin ardından gelen algoritma ise Naive Bayes algoritması oldu. %74 doğruluk değerine ve 0.25 MSE değerine sahip bir şekilde Logistic Regression’u geride bıraktı. Logistic Regression ise %73 doğruluk oranıyla birlikte 0.26 MSE değerini alarak en verimli

beşinci algoritma oldu. Elimizdeki verimlilik düzeyi en düşük olan K-Nearest Neighbor algoritması ile Logistic Regression arasında kalan tek algoritma ise C-Support Vector algoritması. Bu algoritmanın doğruluk değeri %69'ken MSE değeri de 0.3 olarak elimize geçti.

#### IV. SONUÇ

Bu projede kullanılan veri seti sayesinde kırmızı şarabın kalitesinin, ortalamanın üstünde veya altında olduğunu tahmin etmeye çalıştık. Bunu yapmak için, önceden belirlediğimiz algoritmaları kullandık ve sonuçlarını karşılaştırdık. Her algoritmada farklı doğruluk değerleri elde ettik. Bazı algoritmalar bu veri seti için daha yeterli olurken bazıları daha yetersiz kaldı. Algoritmaların veri setiyle verimli çalışabilmesi için, veri setinin detaylandırılması gerekir. Bunu da doğru özniteliklerin eklenmesi veya çıkartılması, örnek sayısının çoğaltılmasıyla sağlayabiliriz. Bu sayede kalite ölçümünü daha efektif hale getirebiliriz.

#### KAYNAKÇA

- [1] DergiPark - Şarap Üretimi ve Kalite  
dergipark.org.tr/sarap-uretimi-ve-kalite
- [2] Wikipedia - Şarap  
wikipedia.org/Şarap
- [3] Resmi Gazete - Türk Gıda Kodeksi Şarap Tebliği  
resmigazete.gov.tr/2009/02/20090204-12
- [4] Kaggle - Red Wine Quality Dataset  
kaggle.com/red-wine-quality
- [5] Wine Quality Exploration  
amazonaws.com/wine-quality-exploration
- [6] Hydrometer Confusion  
creativeconnoisseur.com/hydrometer-confusion
- [7] Erdal Taşçı & Aytuğ Onan - KNN Algorithm  
ab.org.tr/knn-algorithm
- [8] Scholarpedia - K-Nearest Neighbor  
scholarpedia.org/K-nearest\_neighbor
- [9] Erdinç Uzun - KNN Algoritması  
erdincuzun.com/makine\_ogrenmesi/k-nn-algoritmasi
- [10] Large Margin Nearest Neighbor Classification  
proceedings.neurips.cc/Margin\_Nearest\_Neighbor\_Classification.pdf
- [11] DergiPark - Destek Vektör Makinesi  
dergipark.org.tr/Destek\_Vektor\_Makinesi
- [12] Medium - Destek Vektör Makineleri  
medium.com/deep-learning-turkiye/destek-vektor-makineleri
- [13] Medium - Lojistik Regresyon  
medium.com/lojistik-regresyon-makine-ogrenimi
- [14] Başkent Üniversitesi - Naive Bayes  
baskent.edu.tr/Naive\_Bayes.pdf
- [15] Akademik Bilişim - Naive Bayes  
ab.org.tr/Naive\_Bayes.pdf
- [16] Medium - Karar Ağaçları  
medium.com/makine-ogrenimi-karar-agaclar\T1\i
- [17] Sosyal Araştırmalar - Karar Ağacı  
sosyalarastirmalar.com/decision-trees-algorithm
- [18] Medium - Ensemble Learning  
medium.com/ensemble-learning-bagging-ve-boosting
- [19] Medium - Rastgele Orman Ağacı Algoritması  
medium.com/rastgele-orman-algoritmas\T1\i
- [20] DevHunter - Rastgele Orman Ağacı Algoritması  
devhunteryz.com/rastgele-ormanrandom-forest-algoritmasi
- [21] SlideShare - Rastgele Orman Ağacı Algoritması  
slideshare.net/random-forest-algoritmas\T1\i
- [22] Medium - ROC Eğrisi  
medium.com/roc-egrisi