|  |
| --- |
| **Ad-Soyad**: HAKAN ÖZNAM |
| **Proje Başlığı**: Makine Öğrenmesi ile Satranç Oyun Sonu Tahmini |

PROJE 2

**İçindekiler**

[**Projenin Konusu ve Amacı** 2](#_Toc59628387)

[**Projede Kullanılan Materyaller** 3](#_Toc59628388)

[Veri Kümesi ve Özellikleri 3](#_Toc59628389)

[Kullanılan Algoritmalar ve Özellikleri 4](#_Toc59628390)

[**Proje Geliştirme Aşamaları** 7](#_Toc59628391)

[**Proje ile Ulaşılan Sonuçlar** 9](#_Toc59628392)

[LDA kullanmadan ulaşılan sonuçlar 9](#_Toc59628393)

[LDA kullanımı ile ulaşılan sonuçlar 1](#_Toc59628394)3

# **Projenin Konusu ve Amacı**

Bu projede satranç oyunlarının analizi yapılacaktır. [Lichess](https://lichess.org/) adlı satranç sitesinde oynanan 20000’e yakın sayıda maç veri kümesi olarak ele alınmıştır. “Binlerce varyanta sahip satranç oyununda hangi rengin kazanacağını önceden tahmin edebilir miyiz?” Bu sorudan yola çıkılan bu projenin birinci aşamasında tüm veriler analize uygun bir hale getirilmiştir. Projenin ikinci aşamasında makine öğreniminde kullanılan denetimli öğrenme algoritmalarından sınıflandırma algoritmaları test edilecektir.

Sınıflandırma algoritmalarına doğru hiperparametre seçimlerini uyguladıktan sonra algoritmalar kıyaslanacaktır. Daha sonra LDA algoritmasıyla özniteliklerde boyut indirgeme gerçekleştireceğiz. LDA uyguladıktan sonra sınıflandırma algoritmaları bir kez daha kıyaslanacaktır.

Bu projede çeşitli yöntemler kullanarak yapılan sınıflandırmanın sonuçlarda ne gibi değişikliklere yol açacağı gözlemlenmesi amaçlanmıştır.

# **Projede Kullanılan Materyaller**

## Veri Kümesi ve Özellikleri

Proje 1’de ele alınan verilere analize uygun olması amacıyla birtakım işlemler uygulanmıştır. Bu işlemlerden sonra veri kümesi şu şekildedir;

Veri kümesi 9 öznitelik ve 1 etiket sütunu ile 19377 satır veriden oluşmaktadır.

Öznitelikler

**rated** 🡪 Karşılaşmada reyting kazanımı olacak mı? (Evet/Hayır)

**game\_time** 🡪 Karşılaşma süresi

**turns** 🡪 Oyunda geçen hamle sayısı

**victory\_status** 🡪 Oyun sonucu(OneHotEncoder uygulandığı için 4 sütun içermektedir.

**white\_rating** 🡪 Beyaz oyuncu reytingi

**black\_rating** 🡪 Siyah oyuncu reytingi

Etiket

**winner** 🡪 Kazanan taraf veya varsa beraberlik(3 sınıf)

(Black, White, Draw)

## Kullanılan Algoritmalar ve Özellikleri

Projede kullanılan algoritmalara kısaca değinecek olursak;

Grid Search CV

Modelde denenmesi istenen hiperparametreler ve değerleri için bütün kombinasyonlar ile ayrı ayrı model kurulur ve belirtilen metriğe göre en başarılı hiperparametre seti belirlenir.

Logistic Regression

Lojistik regresyon, ikili sonuç veren binary değişkenlerin modellenmesinde kullanılmaktadır. İkili durum söz konusu olan modellerde sonuç genellikle 0 veya 1 olarak tanımlanmaktadır.

Etiket veri kümesinde üç sınıf olduğu için bu algoritma beklediğimiz sonucu vermeyecektir. Fakat yanlış algoritma seçiminde karşımıza çıkabilecek sonuçları görmek amacıyla kullanılacaktır.

Decision Tree Classifier

Karar ağacı, çok sayıda kayıt içeren bir veri kümesini, bir dizi karar kuralları uygulayarak daha küçük kümelere bölmek için kullanılan bir algoritmaların. Yani karar verme adımları uygulanarak, büyük miktarlardaki kayıtları, çok küçük kayıt gruplarına bölerek kullanılan bir algoritmadır. Karar ağaçlarında karşılaşılan en büyük problem overfitting(ezberleme) problemidir.

Random Forest Classifier

Random Forest algoritması, sınıflandırma işlemi esnasında birden fazla karar ağacı üreterek sınıflandırma değerini yükseltmeyi hedefleyen bir algoritmadır. Bireysel olarak oluşturulan karar ağaçları bir araya gelerek karar ormanı oluşturur. Buradaki karar ağaçları bağlı olduğu veri setinden rastgele seçilmiş birer alt kümedir.

K-NN ( K-Nearest Neighbor)

Algoritmanın çalışmasında bir K değeri belirlenir. Bu K değerinin anlamı bakılacak eleman sayısıdır. Bir değer geldiğinde en yakın K kadar eleman alınarak gelen değer arasındaki uzaklık hesaplanır. Uzaklık hesaplama işleminde genelde **Öklid fonksiyonu**kullanılır. Öklid fonksiyonuna alternatif olarak **Manhattan, Minkowski**ve **Hamming**fonksiyonları da kullanılabilir. Uzaklık hesaplandıktan sonra sıralanır ve gelen değer uygun olan sınıfa atanır.

SVM Classifier

Destek vektör makineleriyle sınıflandırmada, iki sınıfa ait örneklerin doğrusal olarak dağıldığını varsayalım. Bu durumda bu iki sınıfın, eğitim verisi kullanılarak elde edilen bir karar fonksiyonu yardımıyla birbirinden ayrılması amaçlanır. Veri setini ikiye ayıran doğru karar doğrusu olarak isimlendirilmektedir. Sonsuz tane karar doğrusu çizebilme imkânı mevcut olsa da önemli olan optimal yani en uygun karar doğrusunu belirlemektir.

Naive Bayes Classifier

Naive Bayes sınıflandırıcısının temeli Bayes teoremine dayanır. **T**embel bir öğrenme algoritmasıdır. Aynı zamanda dengesiz veri kümelerinde de çalışabilir. Algoritmanın çalışma şekli bir eleman için her durumun olasılığını hesaplar ve olasılık değeri en yüksek olana göre sınıflandırır. Az bir eğitim verisiyle çok başarılı işler çıkartabilir.

LDA ( Linear Discriminant Analysis)

Machine learning uygulamaları için preprocessing aşamasında boyut azaltma tekniği olarak kullanılır. Amaç, overfittingi önlemek ve aynı zamanda hesaplama maliyetlerini azaltmaktır. LDA sınıflar arası uzaklığı maksimize etmeye çalışmaktadır. LDA bir gözetimli öğrenme algoritmasıdır.

K-Fold Cross Validation

K-Fold, sınıflandırma modellerinin değerlendirilmesi ve modelin eğitilmesi için veri setini parçalara ayırma yöntemlerinden biridir. K-Fold algoritması veriyi belirlenen bir k sayısına göre eşit parçalara böler, her bir parçanın hem eğitim hem de test için kullanılmasını sağlar, böylelikle dağılım ve parçalanmadan kaynaklanan sapma ve hataları asgariye indirir.

# **Proje Geliştirme Aşamaları**

**1.Aşama (Veri Ön İşleme)**

Proje 1’de elde ettiğimiz verilere ölçeklendirme işlemi ile veriler “0” merkezine indirgendi. Bu sayede matematiksel işlemler daha kısa sürede gerçekleşecektir. Daha sonra verilerimiz öznitelik ve etiket verisi olarak ikiye ayrılmıştır.

**2.Aşama (Parametresiz Modelleme)**

Lojistik regresyon, karar ağaçları, rassal orman, k-en yakın komşu, destek vektör makineleri, naive bayes olmak üzere 6 tane sınıflandırma algoritmasını hiçbir parametre ve hiperparametre kullanmadan çalıştırıldı.

Algoritmalar 5-Fold kullanılarak çalıştırılmıştır. Ölçeklendirilmiş veri üzerinde çalıştırılan tüm modellere ait karmaşıklık matrisi ve sınıflandırma raporu elde edilmiştir.

**3.Aşama (GridSearchCV ile Modelleme)**

GridSearchCV algoritmasıyla birlikte modellerin hiperparametre seçimlerinde değişiklik yapılmıştır. Bu değişim rassal orman, k-en yakın komşu ve destek vektör makineleri modellerine uygulandı. Karşılaştırma için modellere ait karmaşıklık matrisi ve sınıflandırma raporu elde edildi.

**4.Aşama (GridSearchCV ve LDA ile Modelleme)**

3.aşamada modellere uygun hiperparametreler GridSearchCV algoritmasıyla belirlendi. Belirlenen hiperparametrelerle modeller güncellendi. 8 olan öznitelik sayısı LDA ile 2’ye indirgendi.

🡪 Min(öznitelik\_sayısı , grup\_sayısı -1) olduğundan dolayı parametre 2 olarak seçildi.

GridSearchCV ve LDA kullanılan bu aşamada 6 adet model veri kümemize uygulandı. Karmaşıklık matrisi ve sınıflandırma raporu elde edildi.

**5.Aşama (Yeni veri kümesi üzerine Modelleme)**

Bu aşamada öznitelik seçiminde değişiklikler yapıldı. Bu aşamaya kadar olan aşamalarda kullanılan veri kümesi yerine; hamle sayısı(turns) ve oyun sonucu(victory\_status) öznitelikleri seçildi. Bu veri kümesine yine aynı şekil LDA ve GridSearchCV algoritması uygulanan modeller test edildi. Bu modellere ait karmaşıklık matrisi ve sınıflandırma raporu elde edildi.

Tüm aşamalarda elde edilen karmaşıklık matrisi ve sınıflandırma raporu sonuçlar kısmında açıklanacaktır.

# **Proje ile Ulaşılan Sonuçlar**

## LDA kullanmadan ulaşılan sonuçlar

2.aşamada parametresiz modeller uygulandıktan sonra sonuçları incelersek;

| LogisticRegression(random\_state=0) | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,65 | 0,58 | 0,61 | 88931 |
| *Draw* | 0,99 | 0,95 | 0,97 | 910 |
| *White* | 0,65 | 0,71 | 0,68 | 9636 |
| **Accuracy** |  |  | 0,66 | 19377 |

Lojistik regresyon daha çok ikili sınıflarda kullanılan bir algoritma olduğu için doğruluk oranı %66’dır. Modelin “Draw” sonucunu tahmin edebilme skorunu %97 ile yüksek olduğu görülmektedir.

| DecisionTreeClassifier(random\_state=0) | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,58 | 0,58 | 0,58 | 8831 |
| *Draw* | 0,93 | 0,95 | 0,94 | 910 |
| *White* | 0,62 | 0,62 | 0,62 | 9636 |
| **Accuracy** |  |  | 0,62 | 19377 |

Karar ağaçları modeli bu aşamada %62 ile en düşük doğruluk oranına sahip modeldir. “Black” ve “White” tahminleri sırasıyla %58 ve %62 doğruluk oranına sahiptir.

| RandomForestClassifier() | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,67 | 0,66 | 0,66 | 8831 |
| *Draw* | 1,00 | 0,95 | 0,98 | 910 |
| *White* | 0,69 | 0,70 | 0,70 | 9636 |
| **Accuracy** |  |  | 0,69 | 19377 |

Rassal orman sınıflandırma algoritması bu aşamada en yüksek doğruluk oranına(%69) sahip algoritmadır. “Draw” grubunun tahmininde modelin kesinlik oranı %100’dür. Bu oran çok yüksek olduğundan ezberleme problemine yol açabileceği gözlemlenmiştir.

| KNeighborsClassifier() | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,59 | 0,58 | 0,58 | 8831 |
| *Draw* | 0,99 | 0,95 | 0,97 | 910 |
| *White* | 0,62 | 0,62 | 0,62 | 9636 |
| **Accuracy** |  |  | 0,62 | 19377 |

En yakın k komşusuna göre sınıflandırma yapan bu modelde doğruluk oranı %62’dir. Karar ağaçları modelinde olduğu gibi bu model de bu aşamada en düşük doğruluk oranı veren modeldir.

| SVC() | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,65 | 0,56 | 0,60 | 8831 |
| *Draw* | 1,00 | 0,95 | 0,97 | 910 |
| *White* | 0,64 | 0,73 | 0,68 | 9636 |
| **Accuracy** |  |  | 0,66 | 19377 |

| GaussianNB() | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,63 | 0,40 | 0,49 | 8831 |
| *Draw* | 0,96 | 0,95 | 0,96 | 910 |
| *White* | 0,59 | 0,79 | 0,67 | 9636 |
| **Accuracy** |  |  | 0,62 | 19377 |

Bu aşamada son iki model sırasıyla %66 ve %62 doğruluk oranına sahiptir.

Sonuç olarak bu aşamada hiçbir parametre kullanılmadan 6 model denenmiştir. Birçok oyun varyantına sahip satrançta oyunun nasıl sonuçlanacağını tahmin edebilme ortalama olarak %65 oranındadır. Bu oyun üzerinde düşünecek olursak bu oranın yüksek olduğunu söylemek gerekir.

3.aşamada Grid SearchCV algoritması kullanılarak rassal orman, k-en yakın komşu, destek vektör makinelerine en etkili sonuçları veren hiperparametre seçimi uygulanmıştır. Bu hiperparametrelere göre değişimler şu şekildedir.

| RandomForestClassifier(criterion='entropy', n\_estimators=200) | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,67 | 0,66 | 0,66(+0,01) | 8831 |
| *Draw* | 1,00(-0,01) | 0,95 | 0,98(-0,01) | 910 |
| *White* | 0,69 | 0,70(+0,01) | 0,70 | 9636 |
| **Accuracy** |  |  | 0,69(+0,01) | 19377 |

## Rassal orman modelinde hiperparametreler GridSearchCV algoritmasıyla “entropy”,”200” olarak değiştirildi. Parametresiz modele göre doğruluk oranına %1 iyileştirme yapılmıştır.

| KNeighborsClassifier(metric='manhattan', n\_neighbors=21, weights='distance') | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,59(+0,03) | 0,58(+0,02) | 0,58(+0,03) | 8831 |
| *Draw* | 0,99(+0,01) | 0,95 | 0,97(+0,01) | 910 |
| *White* | 0,62(+0,02) | 0,62(+0,05) | 0,62(+0,04) | 9636 |
| **Accuracy** |  |  | 0,62(+0,03) | 19377 |

K-en yakın komşu modelinde GridSearchCV algoritmasıyla birlikte hiperparametreler “manhattan”,”21”,”distance” olarak seçilmiştir. Yeni hiperparametrelere göre doğruluk oranı %3 arttırılmıştır.

| SVC(C=0.75, kernel='linear') | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,65 | 0,56 | 0,60 | 8831 |
| *Draw* | 1,00 | 0,95 | 0,97(+0,01) | 910 |
| *White* | 0,64 | 0,73 | 0,68 | 9636 |
| **Accuracy** |  |  | 0,66 | 19377 |

Destek vektör makineleri algoritmasında hiperparametre seçimi “0,75”,”linear” olarak seçilmiştir. Yeni parametre seçimi doğruluk oranında bir etkiye neden olmamıştır.

## LDA kullanımı ile ulaşılan sonuçlar

LDA kullanıldıktan sonra öznitelik sayısı 9’dan 2 ye indirgenmiştir. Bu işlemin sonuçlar üzerindeki etkisini inceleyecek olursak…

| LogisticRegression(random\_state=0) | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,65 | 0,58 | 0,61 | 88931 |
| *Draw* | 0,99(+0,01) | 0,95 | 0,97 | 910 |
| *White* | 0,65 | 0,71 | 0,68 | 9636 |
| **Accuracy** |  |  | 0,66 | 19377 |

| DecisionTreeClassifier(random\_state=0) | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,58(-0,02) | 0,58(-0,02) | 0,58(-0,02) | 8831 |
| *Draw* | 0,93(+0,02) | 0,95(+0,01) | 0,94(+0,02) | 910 |
| *White* | 0,62(-0,02) | 0,62(-0,02) | 0,62(-0,02) | 9636 |
| **Accuracy** |  |  | 0,62(-0,02) | 19377 |

| RandomForestClassifier(criterion='entropy', n\_estimators=200) | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,67(-0,07) | 0,66(-0,07) | 0,66(-0,06) | 8831 |
| *Draw* | 1,00(-0,03) | 0,95 | 0,98(-0,02) | 910 |
| *White* | 0,69(-0,06) | 0,70(-0,06) | 0,70(-0,07) | 9636 |
| **Accuracy** |  |  | 0,69(-0,06) | 19377 |

| KNeighborsClassifier(metric='manhattan', n\_neighbors=21, weights='distance') | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,59(+0,02) | 0,58(+0,02) | 0,58(+0,03) | 8831 |
| *Draw* | 0,99 | 0,95 | 0,97 | 910 |
| *White* | 0,62(+0,02) | 0,62(+0,04) | 0,62(+0,03) | 9636 |
| **Accuracy** |  |  | 0,62(+0,02) | 19377 |

| SVC(C=0.75, kernel='linear') | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,65 | 0,56 | 0,60 | 8831 |
| *Draw* | 1,00 | 0,95 | 0,97(+0,01) | 910 |
| *White* | 0,64 | 0,73 | 0,68(+0,01) | 9636 |
| **Accuracy** |  |  | 0,66 | 19377 |

| GaussianNB() | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,63(+0,01) | 0,40(+0,19) | 0,49(+0,12) | 8831 |
| *Draw* | 0,96(+0,04) | 0,95 | 0,96(+0,01) | 910 |
| *White* | 0,59(+0,06) | 0,79(-0,07) | 0,67(+0,01) | 9636 |
| **Accuracy** |  |  | 0,62(+0,04) | 19377 |

LDA ile boyut indirgedikten sonra modellerin doğruluk oranlarında çok az bir değişim gözlemlenmiştir. 9 öznitelik ile elde edilen doğruluk sonucunu LDA ile 2 sütunda yaklaşık olarak elde ettik. Büyük veri kümelerinde runtime ve depolama alanının efektif kullanımı önemli olduğundan dolayı LDA kullanmanın gerekliliği gösterilmiştir.

Son olarak yeni elde edilen veri kümesine modelleri uygulayıp sonuçlara bakalım.

| LogisticRegression(random\_state=0) | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,49 | 0,17 | 0,26 | 8831 |
| *Draw* | 0,99 | 0,95 | 0,97 | 910 |
| *White* | 0,53 | 0,84 | 0,65 | 9636 |
| **Accuracy** |  |  | 0,54 | 19377 |

| DecisionTreeClassifier(random\_state=0) | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,89 | 0,90 | 0,89 | 8831 |
| *Draw* | 0,99 | 0,96 | 0,97 | 910 |
| *White* | 0,90 | 0,90 | 0,90 | 9636 |
| **Accuracy** |  |  | 0,90 | 19377 |

| RandomForestClassifier(criterion='entropy', n\_estimators=200) | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,89 | 0,90 | 0,89 | 8831 |
| *Draw* | 0,98 | 0,96 | 0,97 | 910 |
| *White* | 0,90 | 0,90 | 0,90 | 9636 |
| **Accuracy** |  |  | 0,90 | 19377 |

| KNeighborsClassifier(metric='manhattan', n\_neighbors=21, weights='distance') | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,89 | 0,90 | 0,89 | 8831 |
| *Draw* | 0,99 | 0,96 | 0,97 | 910 |
| *White* | 0,91 | 0,90 | 0,90 | 9636 |
| **Accuracy** |  |  | 0.90 | 19377 |

| SVC(C=0.75, kernel='linear') | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,41 | 0,01 | 0,02 | 8831 |
| *Draw* | 1,00 | 0,95 | 0,98 | 910 |
| *White* | 0,52 | 0,99 | 0,68 | 9636 |
| **Accuracy** |  |  | 0,54 | 19377 |

| GaussianNB() | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| *Black* | 0,50 | 0,21 | 0,30 | 8831 |
| *Draw* | 0,99 | 0,95 | 0,97 | 910 |
| *White* | 0,53 | 0,81 | 0,64 | 9636 |
| **Accuracy** |  |  | 0,54 | 19377 |

Yeni veri kümesini modellere uyguladığımızda modeller arasında %36 oranında fark görülmektedir. Karar ağaçları, rassal orman, k-en yakın komşu modellerinde %90 doğruluk sonucu elde edilmiştir. Aynı zamanda diğer algoritmalar olan lojistik regresyon, destek vektör makineleri ve naive bayes modelleri %54 doğruluk oranında kalmıştır.

Lojistik regresyon ve destek vektör makineleri daha çok ikili sınıflarda yüksek başarı göstermektedir. Bu nedenle bu projede her aşamada diğer modellere göre düşük doğruluk oranı göstermektedir. Satranç oyunu oyun sonunun tahmininin zor olduğu bir mücadeledir. Buna rağmen kullandığımız modellerin yüksek doğruluk oranı verdiği gözlemlenmiştir.

Sonuç olarak;

Bir satranç sitesinden yaklaşık 20000 müsabaka veri kümesi olarak seçildi. Seçilen veri kümesi ön işleme aşamalarından geçirilerek sınıflandırma algoritmaları uygulanacak düzeye getirildi. Bu aşamada oyun sonunu ne kadar doğru tahmin edilebileceğini test etmek üzere modeller uygulandı ve sonuçlar karşılaştırıldı.

Her şeye rağmen satranç dâhisi Booby Fischer’ın dediği gibi;

“Neredeyse kesin olarak, oyun teorik açıdan beraberedir”