İÇİNDEKİLER

1. ÖZET

Futbol, dünya genelindeki taraftar sayısı açısından en popüler sporlardan biridir. Bu durum, futbolun öngörülemez doğasından kaynaklanmaktadır. İnsanlar, heyecan ve sevinç gibi duyguları bir araya getiren bu sporla giderek daha fazla bağlantı kurmaktadır. Maç sonucu tahmini çok zor bir problem olup, bu problem için çözümler son zamanlarda oldukça popüler hale gelmiştir. Bu öngörülemeyen oyunun sonucuyla ilgili maç sırasında gerçekleşen etkilerin makine öğrenimi yöntemleriyle tahmin edilmeye çalışılmaktadır. Bu çalışma, İngiltere Premier League liginin 2017-2018 ile 2023-2024 sezonları arasındaki 2449 maçtan elde edilen maç istatistiklerini kullanarak maç sonucu tahmininde en etkili hangi modelin daha başarılı olduğunu belirlemek üzere yapılmıştır. Her maç için 46 özellik bulunmaktadır. Bu özellikler kullanılarak yapılan testlerde en iyi sonuçları "Neural net (85.70%), "Logistic Regression (83.74%)" ve "Gradient Boost Tree (82,48%)" modelleri vermiştir. Model eğitimi için kullanılan özelliklerin ağırlıklarına göre sıralanıp modellerdeki değerlerinin ortalamaları alınarak yeni özellik kümesi ile yapılan modelleme testlerinde Neural net modelinin doğruluk yüzdesinin yükseldiği ve diğer iki modelin doğruluk yüzdelerinin düştüğü gözlemlenmiştir.

2. KULLANILAN PROGRAM

RapidMiner, veri madenciliği, makine öğrenimi ve veri analizi gibi işlemleri gerçekleştirmek için kullanılan açık kaynaklı bir veri bilimi platformudur.

Genel olarak iş ve ticari uygulamalarda kullanıldığı gibi aynı zamanda araştırma, eğitim, hızlı prototipleme ve uygulama geliştirme gibi amaçlarla da kullanılabilir. Ayrıca, veri madenciliği sürecinin tüm adımları RapidMiner tarafından desteklenmektedir, bu yüzden veri hazırlama, sonuçları görselleştirme, doğrulama ve optimizasyon gibi amaçlarla da yazılımın kullanılması mümkündür. RapidMiner ile sadece birkaç tıklamayla tüm verilerinize erişebilir ve bunları hazırlayabilir, en iyi modeli oluşturabilir ve üretime geçirebilirsiniz. [1]

3. SINIFLANDIRMA ALGORİTMALARI

Bu bölümde sınıflandırma nedir ve niçin kullanılır birkaç cümleyle anlatınız. Hangi programı kullanacaksanız bu algoritmalardan 3 tanesini belirtiniz.

3.1. Logistic Regression

Logistic Regression, Makine öğreniminde, ikili bir sonucun olasılığını belirlemek için doğrusal regresyonun bir alt kümesi olan lojistik regresyon kullanılır. bu model yalnızca iki sınıfa (binominal) ayrılacak verilerde kullanılabilir. Fakat üç veya daha fazla sınıflı verilerde bu algoritmayı kullanmak için her bir sınıfı kalan diğer sınıflarla karşılaştırıp ortalamasını alırız. Doğrusal bir modeldir, giren ve çıkanlar arasındaki basit ilişkiye dayalı olasılık tahmini yapar. [2][6]

3.2. Neural Network

İnsan beyninin anatomisine dayanan bir makine öğrenme modelidir. Bilgi, ağ bağlantılı nöron katmanları tarafından işlenir ve gönderilir. Hem regresyon hem de sınıflandırma tahminlerinde kullanılabilir. Veriler arasındaki karmaşık bağlantıları bulabilir ve soyut yorumlarda bulunabilir. Doğruluk yüzdesi veri setinin büyüklüğü ile doğru orantılıdır. Küçük veri setlerinde hatalı sonuçlar verebilir.[3][6]

3.3. Gradient Boosted Trees

Zayıf tahmin modellerini birbirine birleştirerek sonuca giden birleşik bir modeldir. Bu zayıf modeller veri hakkında yalnızca birkaç çıkarım yapabilenlerdir, “Decision Tree” gibi. Ayrıca performansı ile “Random Forest” modelini de geri de bırakmaktadır. Doğrusal olmayan verilerin ilişkilerini öğrenmede etkilidir. [4]

4. MATERYAL VE YÖNTEM

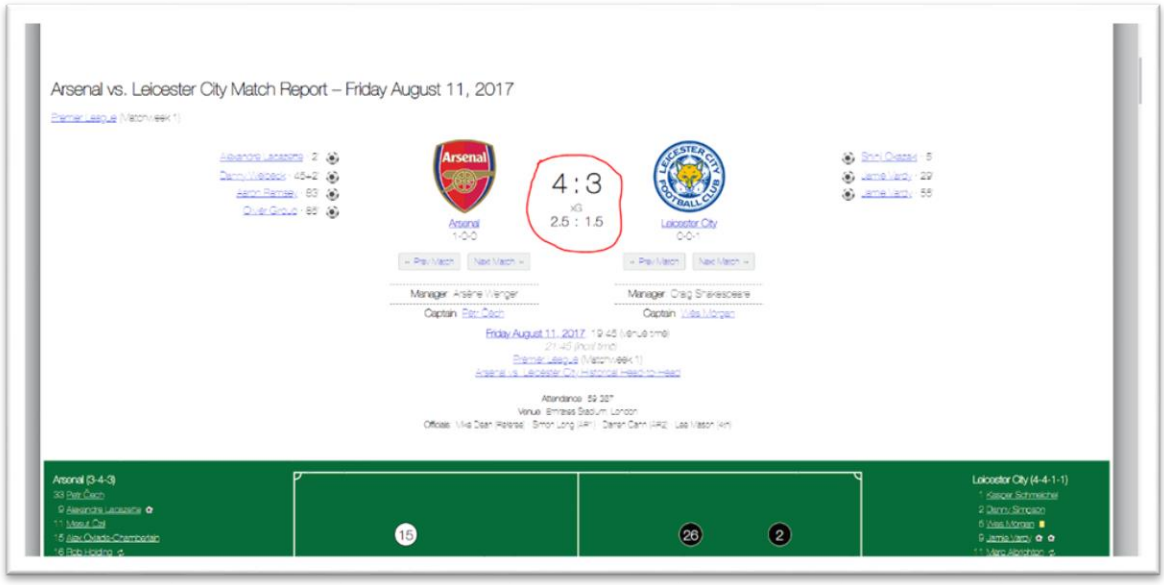
4.1. Verinin Elde Edilmesi

Veriler, fbref.com sitesinden Python programlama dili ve kütüphaneleri olan “requests” ve “beautifulsoup” ile web scraping yaparak elde edildi. Verileri elde etmek için öncellikle verilerin bulunduğu 2448 web sayfasının linkleri; "2017'den itibaren her sene 38 hafta boyunca oynanan tüm maçlar ve bu sene oynanan maçlar, üç farklı algoritma kullanılarak bir txt dosyasına kaydedildi. (Resim 4.1.1, Resim 4.1.2, Resim 4.1.3) Bu web sayfalarının html dosyaları web site sunucusunu gereksiz meşgul etmemek ve olası internet kesintilerinde sakınmak için bilgisayara indirildi. Böylece veriler hep elimizin altında bulundu. (Resim 4.1.4, Resim 4.1.5)

Veriler sayfada metin şeklinde ve statik halde bulunduğundan verilerin html kodlarındaki yerleri “html-selector” özellikleri ile referanslanıp bir txt dosyasına kaydedildi.(Resim 4.1.1, Resim 4.1.6) Daha sonra bu dosyadaki referanslar kullanılarak hazırlanan algoritma yardımı ile bütün html dosyalarındaki her biri veri türü isminin bulunduğu bir txt dosyasına kaydedildi. Bu yöntemle elde edilemeyen 20 veri türü farklı manuel algoritmalar ile aynı çeşit txt dosyalarına kaydedildi.

Bunun sonucunda 48 adet verilerin çeşidine göre isimlendirilmiş txt dosyası elde edildi. Elde edilen bu txt dosyaları kullanılarak yine bir Python kütüphanesi olan “pandas” ile tüm veri ayrıntılarını içeren bir Excel dosyası elde edildi. Toplamda 2448 veri ve 48 özellik içeren bir veri setine ulaştık. (Resim 4.1.7)

Elde edilen verilerde eksiklikler (Resim 4.1.8) ve fazlalıklar (Resim 4.1.9) vardı. Bazı verilerin ise türleri farklıydı ve tür dönüşümü yapılmalıydı (Resim 4.1.10). Ayrıca sınıflandırmayı içeren sütunu da verilerdeki “T1_score” ve “T2_score” sütunlarından elde etmemiz gerekiyordu (Resim 4.1.11). Bunlar için de “Rapidminer” programının “Turbo Prep” özelliğinden yararlandık.



Resim 4.1.1: *Verilerin Çekildiği fbref.com sitesinin maç sayfası.*

Resim 4.1.2: *Verilerin çekildiği fbref.com sitesinin maç sayfasının istatistik verileri.*

```python
from bs4 import BeautifulSoup
import os
from concurrent.futures import ProcessPoolExecutor
from itertools import product

def read_output_file_names_and_selectors(file_path):
    with open(file_path, 'r') as file:
        lines = [line.strip() for line in file.readlines() if line.strip()]


        if len(lines) % 2 == 0:
            data = {lines[i]: lines[i+1] for i in range(0, len(lines), 2)}
        else:
            print("Error: Incomplete pairs of output file names and selectors.")
            data = {}

        return data

def process_html_file(directory, file_name, selector):
    extracted_texts = []
    file_path = os.path.join(directory, file_name)

    try:
        with open(file_path, 'r', encoding='utf-8') as html_file:
            soup = BeautifulSoup(html_file, 'html.parser')
            elements = soup.select(selector)

            if elements:
                for element in elements:
                    text = element.get_text(strip=True)
                    extracted_texts.append(text)
            else:
                print(f"No elements found in {file_name} with the provided selector.")

    except Exception as e:
        print(f"Error processing {file_name}: {e}")

    return extracted_texts

def extract_text_from_elements(directory, data):
    html_directory = directory
    output_directory = 'data_individuals'

    if not os.path.exists(output_directory):
        os.makedirs(output_directory)

    html_files = [f for f in os.listdir(html_directory) if f.endswith('.html')]

    for output_file, selector in data.items():
        selector_html_pairs = product([selector], html_files)
        batch_size = 420

        for batch in [list(batch) for batch in zip(*[iter(selector_html_pairs)]*batch_size)]:
            with ProcessPoolExecutor() as executor:
                futures = [executor.submit(process_html_file, html_directory, file_name, sel)
for sel, file_name in batch]
                extracted_texts = []

                for future in futures:
                    result = future.result()
                    extracted_texts.extend(result)

                output_file_path = os.path.join(output_directory, f'{output_file}.txt')
                try:
                    with open(output_file_path, 'a', encoding='utf-8') as output:
                        for text in extracted_texts:
                            output.write(text + '\n')
                    print(f"Text extracted and saved to {output_file_path}")

                except Exception as e:
                    print(f"Error writing to the output file: {e}")

if __name__ == "__main__":

    html_directory = 'matches_htmls'
    output_file_path = 'data_selectors.txt'

    data = read_output_file_names_and_selectors(output_file_path)
    extract_text_from_elements(html_directory, data)
```

Resim 4.1.3: *Sitenin en alt kısmında kalan verileri çeken kod.*

```python
from re import A
from bs4 import BeautifulSoup
import requests

req_url = "https://fbref.com/en/comps/9/schedule/Premier-League-Scores-and-Fixtures"

req = requests.get(req_url)

soup = BeautifulSoup(req.text, "html.parser")

# Table seçimini daha önceki satırdan ayır
scorboard_table = soup.select('table.stats_table')[0]

# Table içindeki tüm td.center etiketlerini al
scorboard_cells = scorboard_table.find_all('td', class_='center')

# Her td.center içindeki a etiketlerini topla
teams_tag = [cell.find('a') for cell in scorboard_cells if cell.find('a')]

# a etiketlerinin href özniteliklerini al
teams_tag = [l.get("href") for l in teams_tag if l]

# /matches/ içeren href'leri filtreleyerek takım URL'lerini oluştur
etikets = [l for l in teams_tag if '/matches/' in l]
takimlar_urls = [f"https://fbref.com{l}" for l in etikets]
with open('premier_urls.txt', 'w') as file:
    a =1
    for url in takimlar_urls:
        file.write(url + '\n')
        a=a+1

print(a)
```

Resim 4.1.4: *Sitedeki uygun olan bütün maçların linkini bir .txt dosyasına kaydeden kod.*

```python
import os
import requests
import time

def download_web_pages(file_path, output_directory, start_index):
    # Create the directory if it doesn't exist
    if not os.path.exists(output_directory):
        os.makedirs(output_directory)

    # Read links from the file
    with open(file_path, 'r') as file:
        links = file.read().splitlines()

    # Download web pages
    for i, link in enumerate(links, start=start_index):
        try:
            response = requests.get(link)
            if response.status_code == 200:
                # Save the HTML content to a file
                filename = f"{i:03}.html"
                filepath = os.path.join(output_directory, filename)
                with open(filepath, 'w', encoding='utf-8') as html_file:
                    html_file.write(response.text)
                print(f"Downloaded: {filename}")
                time.sleep(2)
            else:
                print(f"Failed to download: {link} - Status code: {response.status_code}")
        except Exception as e:
            print(f"Failed to download: {link} - {e}")

# File paths
links_file = 'premier17-23_urls.txt'
output_dir = 'premier_html'

# Download web pages starting from index 301
download_web_pages(links_file, output_dir, start_index=402)
```

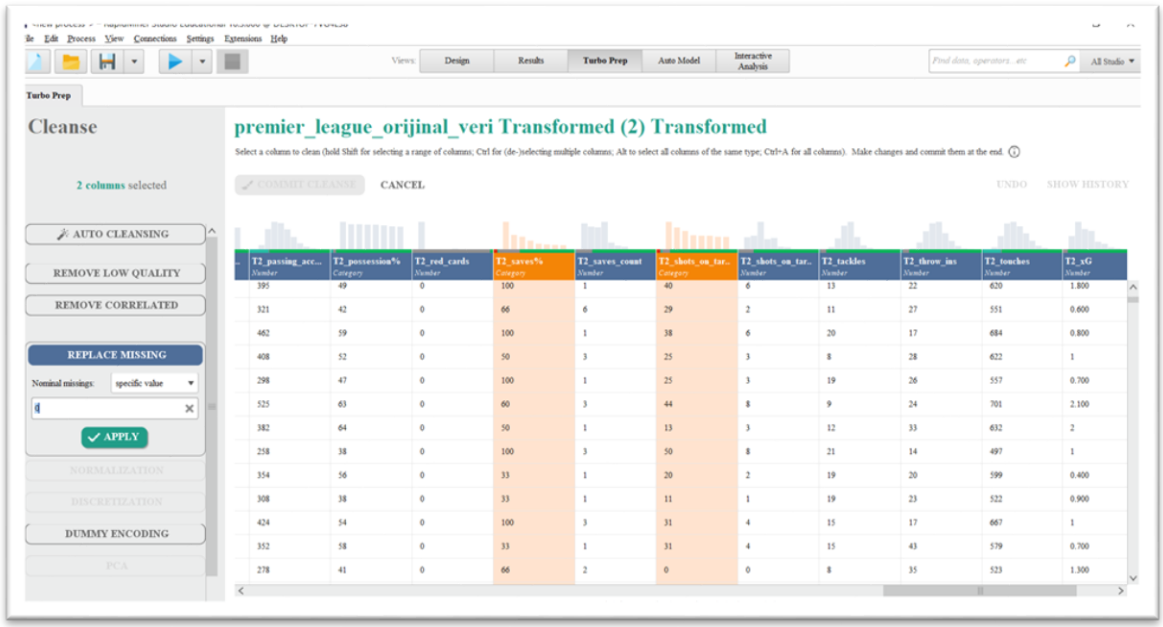
Resim 4.1.5: *.txt dosyasına kaydedilmiş olan linklerin .html dosyalarını bir klasöre indiren kod.*

```
T1_score
#content > div.scorebox > div:nth-child(1) > div.scores > div.score

T1_xG
#content > div.scorebox > div:nth-child(1) > div.scores > div.score_xg

T1_fouls
#team_stats_extra > div:nth-child(1) > div:nth-child(4)

T1_corners
#team_stats_extra > div:nth-child(1) > div:nth-child(7)

T1_crosses
#team_stats_extra > div:nth-child(1) > div:nth-child(10)

T1_touches
#team_stats_extra > div:nth-child(1) > div:nth-child(13)

T1_tackles
#team_stats_extra > div:nth-child(2) > div:nth-child(4)

T1_interceptions
#team_stats_extra > div:nth-child(2) > div:nth-child(7)
```

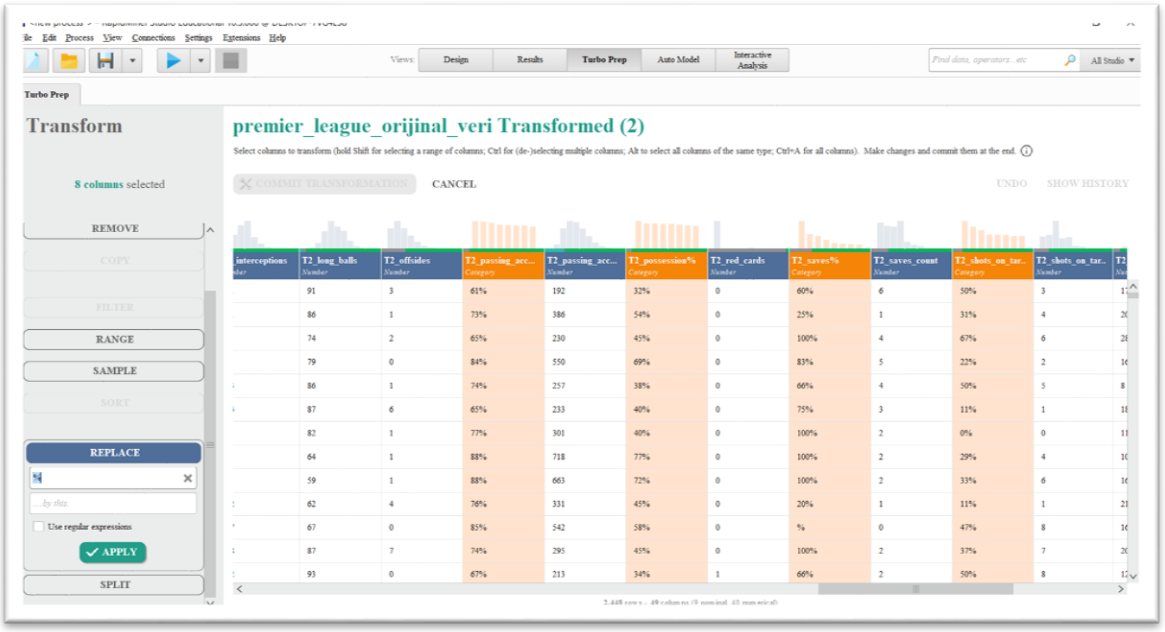
Resim 4.1.6: *html-selector dosyasının içeriğinin bir kısmı.*

```python
import pandas as pd
import os
from openpyxl.workbook import Workbook


folder_path = 'all_datas'


files = [f for f in os.listdir(folder_path) if f.endswith('.txt')]


data = pd.DataFrame()


for file in files:
    file_name = os.path.splitext(file)[0]
    file_data = pd.read_csv(os.path.join(folder_path, file), header=None)
    data[file_name] = file_data[0]


excel_file_path = 'football_final.xlsx'
data.to_excel(excel_file_path, index=False)
```

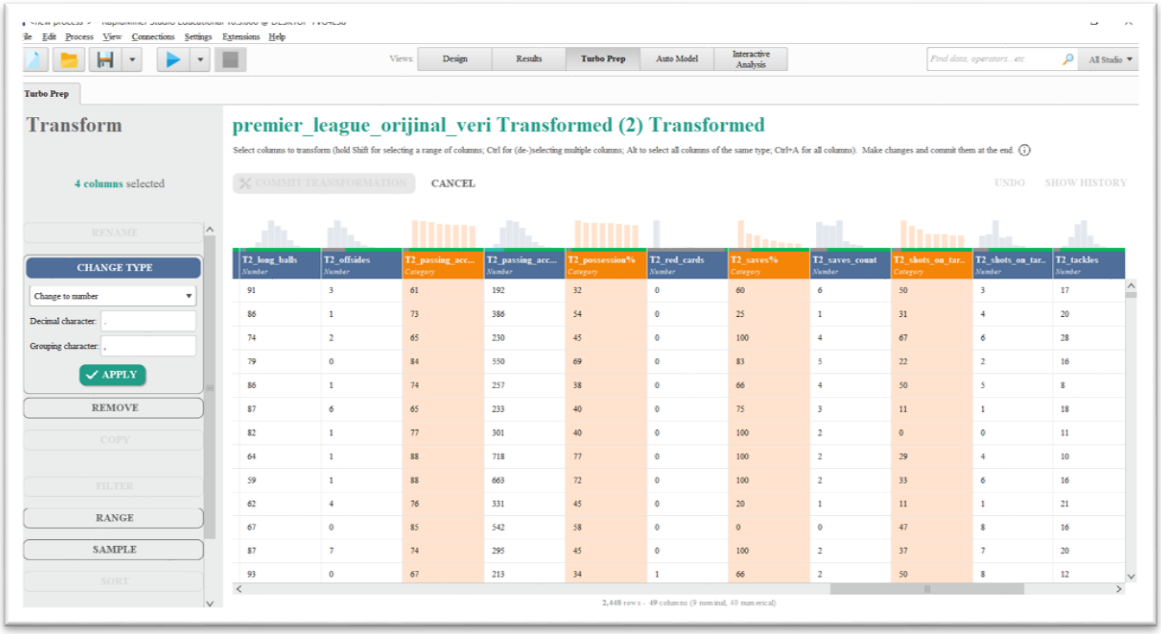
Resim 4.1.7: *.txt dosyalarında bulunan verileri .xlsx dosyasına çeviren kod.*



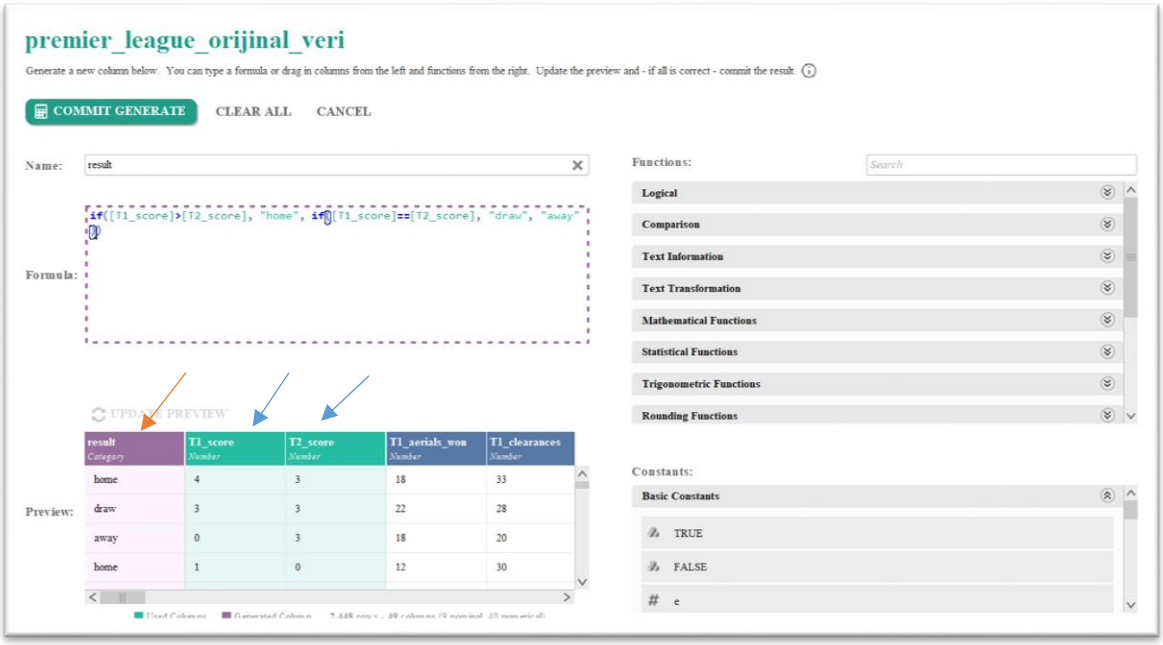
Resim 4.1.8: *Bazı "0" değerine sahip verilerin site kaynaklı sorundan elde edilememesi nedeniyle eksikliklerinin tamamlanması.*



Resim 4.1.9: *Bazı verilerdeki, gereksiz, "%" işaretlerinin kaldırılması.*



Resim 4.1.10: *"%" işaretinden dolayı program tarafından yanlış yapılan ve model eğitimlerine sıkıntı çıkaracak kategorilendirmelerin düzeltilmesi.*



Resim 4.1.11: *İki farklı sütundaki verilerin karşılaştırılması ile sınıflandırma sütunun doğru şekilde oluşturulması.*

4.2. Verinin Özellikleri

Bu çalışmada elde edilen özellikler bir futbol maçındaki her iki takımın da sahip olduğu istatiksel verilerdir. Bütün veriler sayı şeklinde olduğundan herhangi bir dönüşüm yapılmamıştır. Veri sınıfı ise T1_score ve T2_score özelliklerinin birbirleri ile karşılaştırılmalarından yararlanılarak üretilmiştir. T1_score değeri büyükse "home", küçükse "away" ve eşit ise "draw" kelimesi üretilmiştir.

Özellikler her iki takım için de aynı olduğundan takım ayrımı olmadan yazılacaktır. Ayrıca çok fazla değer içerdiklerinden aralık olarak yazılacaklardır.

Veriler; "*home*", "*draw*" ve "*away*" olarak üç sınıfa ayrılmıştır.

Özellik İsmi	**Özellik Açıklaması**	**Değer Aralığı (min - max)**
Aerials Won	Kazanılan hava toplarının sayıları	1- 65
Clearances	Topu uzaklaştırma sayıları	1 - 80
Corners	Korner atış sayıları	0 – 19
Crosses	Yapılan orta sayıları	0 – 64
Fouls	Yapılan faul sayıları	0 – 26
Goal Kicks	Kale vuruşu sayıları	0 – 23
Interceptions	Pas arası yapma sayıları	0 – 33
Long Balls	Uzun pas sayıları	25 – 127
Offsides	Ofsayt durumunda kalma sayıları	0 – 10
Passing Accuracy %	İsabetli pas yüzdeleri	46 – 92
Passing Accuracy Count	İsabetli pas sayıları	84 – 957
Possession %	Topa sahip olma yüzdeleri	18 – 82
Red Cards	Kırmızı kart sayıları	0 – 2
Saves %	Kurtarış yüzdeleri	0 – 100
Saves Count	Kurtarış sayıları	0 – 13
Score	Atılan gol sayıları	0 – 9

Shots on Target %	İsabetli şut yüzdeleri	0 – 100
Shots on Target Count	İsabetli şut sayıları	0 – 15
Tackles	Top çalma sayıları	3 – 46
Throw-ins	Taç atışı sayıları	3 – 44
Touches	Topla bulaşma sayıları	297 – 1120
xG	Gol olabilecek pozisyon hesaplamaları	0 – 5.9
Yellow Cards	Sarı kart sayıları	0 – 7
Yellow to Red Cards	İki sarı kart sonrası kırmızı kart sayıları	0 – 1

4.3. Karmaşıklık Matrisi ve Performans Ölçümleri

Karmaşıklık matrisi, bir sınıflandırma modelinin performansını değerlendirmek için kullanılır. Makine öğrenmesinde kullanılan sınıflandırma modellerinin performansını değerlendirmek için hedef niteliğe ait tahminlerin ve gerçek değerlerin karşılaştırıldığı hata matrisi sıklıkla kullanılmaktadır. İki boyutlu bir matristir ve gerçek sınıfları (satırlar) ve tahmin edilen sınıfları (sütunlar) içerir.[5]

	Gerçek Sınıf 0	Gerçek Sınıf 1	Gerçek Sınıf 2
Tahmin Sınıf 0	TP0	FP0	FP0
Tahmin Sınıf 1	FP1	TP1	FP1
Tahmin Sınıf 2	FP2	FP2	TP2

TP: True Positive (Doğru Pozitif)

FP: False Positive (Yanlış Pozitif)

TN: True Negative (Doğru Negatif)

FN: False Negative (Yanlış Negatif)

Çok sınıflı sınıflandırma performans ölçümleri her bir sınıf için ayrı ayrı hesaplanır ve ardından ortalama değerleri alınır.

Ortalama Accuracy (%)

Accuracy = ((TP + TN) / (TP + FP + TN +FN)) *100

Ortalama Precision (%)

Precision = (TP/ (TP + FP)) *100

Ortalama Recall (%)

Recall = (TP/ (TP + FN)) *100

Ortalama F1-Skor (%)

F1-Skor = ((2* Precision * Recall) / (Precision + Recall)) * 100

4.4. Çapraz Doğrulama

Cross validation, bir modelin, pratikte, ne kadar doğru sonuç verip vermediğini kontrol etmek için yapılan bir test ve uygulama sürecidir. Aşağıdaki gibi 6 adımda uygulanabilir.

1-) Veriyi ele alma:

2-) Veriyi “k” adet eşit subsete bölme (bu örnekte k=10):

3-) Herhangi bir subseti model test etmek için seçme:

4-) Kalan diğer subsetleri model eğitimi için kullanma:

5-) Oluşan modeli test etme:

6-) Subseti değiştirip k kadar aynı işlemleri tekrarlama:

5. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

5.1. Logistic Regression Sonuçları

Logistic Regression algoritmasından elde edilen sonuçlarda, toplam 2248 veriden 2056 tanesi doğru sınıflandırılmıştır yani yapılan 2248 tahminin 2056 tanesi doğrudur. Tablo 5.1.1'de True home (gerçek doğru) sınıfını incelendiğinde 1096 verinin 997 tanesinin doğru sınıflandırılmıştır doğruluk oranı %89.96, duyarlılık oranı %90.97, kesinlik oranı %89.10 ve f1-skor oranı %90.02'dir. Algoritmanın ev sahibi takımın kazanma durumunu doğru tahmin ettiği durum sayısı oldukça yüksektir. Benzer şekilde True away (gerçek doğru) sınıfını incelediğimizde 800 verinin 716 tanesinin doğru sınıflandırılmıştır doğruluk oranı %89.5, duyarlılık oranı %89.50, kesinlik oranı %87.42 ve f1-skor oranı %88.44'tür. Algoritmanın deplasman takımın kazanma durumunu doğru tahmin ettiği durum sayısı oldukça yüksektir ama ev sahibi tahmininden düşüktür. Son olarak True draw (gerçek doğru) sınıfını incelediğimizde 552 verinin 337 tanesinin doğru sınıflandırılmıştır doğruluk oranı %66.05, duyarlılık oranı %61.05, kesinlik oranı %66.08 ve f1-skor oranı %63.46'dır Algoritmanın berabere kalma durumunu doğru tahmin etme oranı, genel olarak ev sahibi takımın ve deplasman takımının kazanma tahmini oranından oldukça düşüktür. Bu durum, ortalama bir doğruluk seviyesi olarak değerlendirilebilir.

Tablo 5.1.1'de Logistic Regression algoritmasına ait karmaşıklık matrisi verilmiştir. Karmaşıklık matrisinden Tablo 5.1.2 elde edilmiştir. Tablo 5.1.2 incelendiğinde ortalama doğruluk oranı %83.74'dir bu oranının yüksek olması, modelin genel olarak başarılı bir şekilde sınıflandırma yaptığını gösterir. Ortalama duyarlılık oranı %80.50'dır bu oran modelin pozitif durumları atlamadan tanıma konusunda iyi bir performans sergilediğini gösterir. Ort. kesinlik oranı %80.86'dir bu oran modelin pozitif olarak sınıflandırdığı durumların çoğunun gerçekten pozitif olduğunu gösterir.

Tablo 5.1.1.

Team		Tahmini Sınıf		
		Pred Home	**Pred Away**	**Pred Draw**
Gerçek Sınıf	**True Home**	**997**	**3**	**96**
	True Away	**7**	**716**	**77**
	True Draw	**115**	**100**	**337**

Tablo 5.1.2.

Performans Sonuçları	Yüzde (%)
Ort. Accuracy (%)	83.74
Ort. Precision (%)	80.86
Ort. Recall (%)	80.50
Ort. F1-Skor (%)	80.64

5.2. Neural Net Sonuçları

Neural Net algoritmasından elde edilen sonuçlarda, toplam 2248 veriden 2098 tanesi doğru sınıflandırılmıştır yani yapılan 2248 tahminin 2098 tanesi doğrudur. Tablo 5.2.1'de True home (gerçek doğru) sınıfını incelendiğinde 1096 verinin 1045 tanesinin doğru sınıflandırılmıştır doğruluk oranı %95.34, geri duyarlılık %95.35, kesinlik oranı %87.59 ve f1-skor oranı %91.30'dir. Algoritmanın ev sahibi takımın kazanma durumunu doğru tahmin ettiği durum sayısı oldukça yüksektir. Benzer şekilde True away (gerçek doğru) sınıfını incelediğimizde 800 verinin 754 tanesinin doğru sınıflandırılmıştır doğruluk oranı %94.25, duyarlılık oranı %94.25, kesinlik oranı %86.87 ve f1-skor oranı %90.40'dür. Algoritmanın deplasman takımın kazanma durumunu doğru tahmin ettiği durum sayısı oldukça yüksektir ama ev sahibi tahmininden düşüktür Son olarak berabere tahmini sınıfını incelediğimizde 552 verinin 299 tanesinin doğru sınıflandırılmıştır doğruluk oranı %54.16, duyarlılık oranı %54.17, kesinlik oranı %77.26 ve f1-skor oranı %63.68'dır. Algoritmanın berabere kalma durumunu doğru tahmin etme oranı, genel olarak ev sahibi takımın ve deplasman takımının kazanma tahmini oranından oldukça düşüktür. Bu durum, ortalama bir doğruluk seviyesi olarak değerlendirilebilir.

Tablo 5.2.1'de Neural net algoritmasına ait karmaşıklık matrisi verilmiştir. Karmaşıklık matrisinden Tablo 5.2.2 elde edilmiştir Tablo 5.2.2 incelendiğinde ortalama doğruluk oranı %85.70'dir bu oranının yüksek olması, modelin genel olarak başarılı bir şekilde sınıflandırma yaptığını gösterir. Ortalama duyarlılık oranı %%78.25'dir bu oran modelin pozitif durumları atlamadan tanıma konusunda iyi bir performans sergilediğini gösterir. Ort. kesinlik oranı %83.90'dır bu oran modelin pozitif olarak sınıflandırdığı durumların çoğunun gerçekten pozitif olduğunu gösterir.

Tablo 5.2.1.

Team		Tahmini Sınıf		
		Home	**Away**	**Draw**
Gerçek Sınıf	**Home**	**1045**	**5**	**46**
	Away	**4**	**754**	**42**
	Draw	**144**	**109**	**299**

Tablo 5.2.2.

Performans Sonuçları	Yüzde (%)
Ort. Accuracy (%)	85.70
Ort. Precision (%)	83.90
Ort. Recall (%)	78.25
Ort. F1-Skor (%)	81.79

5.3. Gradient Boosted Trees Sonuçları

Gradient boosted trees algoritmasından elde edilen sonuçlarda, toplam 2248 veriden 2019 tanesi doğru sınıflandırılmıştır yani yapılan 2248 tahminin 2019 tanesi doğrudur. Tablo 5.3.1'de Ev sahibi tahmini sınıfını incelendiğinde 1096 verinin 1002 tanesinin doğru sınıflandırılmıştır doğruluk oranı %91.42, duyarlılık oranı %91.42, kesinlik oranı %87.13 ve f1-skor oranı %89.22'dir. Algoritmanın ev sahibi takımın kazanma durumunu doğru tahmin ettiği durum sayısı oldukça yüksektir. Benzer şekilde deplasman takımı tahmini sınıfını incelediğimizde 800 verinin 695 tanesinin doğru sınıflandırılmıştır doğruluk oranı %86.87, duyarlılık oranı %86.88, kesinlik oranı %84.45 ve f1-skor oranı %85.64'dür. Algoritmanın deplasman takımın kazanma durumunu doğru tahmin ettiği durum sayısı oldukça yüksektir ama ev sahibi tahmininden düşüktür Son olarak berabere tahmini sınıfını incelediğimizde 552 verinin 322 tanesinin doğru sınıflandırılmıştır doğruluk oranı %58.33, duyarlılık oranı %58.33, kesinlik oranı %67.79 ve f1-skor oranı %62.70'dır. Algoritmanın berabere kalma durumunu doğru tahmin etme oranı, genel olarak ev sahibi takımın ve deplasman takımının kazanma tahmini oranından oldukça düşüktür. Bu durum, ortalama bir doğruluk seviyesi olarak değerlendirilebilir.

Tablo 5.3.1'de Gradient boosted trees algoritmasına ait karmaşıklık matrisi verilmiştir. Karmaşıklık matrisinden Tablo 5.3.2 elde edilmiştir. Tablo 5.3.2 incelendiğinde ortalama doğruluk oranı %82.48'tir bu oranının yüksek olması, modelin genel olarak başarılı bir şekilde sınıflandırma yaptığını gösterir. Ortalama duyarlılık oranı %%78.77'tir bu oran modelin pozitif durumları atlamadan tanıma konusunda iyi bir performans sergilediğini gösterir. Ort. kesinlik oranı %79.79'dır bu oran modelin pozitif olarak sınıflandırdığı durumların çoğunun gerçekten pozitif olduğunu gösterir.

Tablo 5.3.1.

Team		Tahmini Sınıf		
		Home	**Away**	**Draw**
Gerçek Sınıf	**Home**	**1002**	**22**	**72**
	Away	**24**	**695**	**81**
	Draw	**124**	**106**	**322**

Tablo 5.3.2.

Performans Sonuçları	Yüzde (%)
Ort. Accuracy (%)	82.48
Ort. Precision (%)	79.79
Ort. Recall (%)	79.77
Ort. F1-Skor (%)	78.85

6. SONUÇLAR

6.1. Doğruluk (Accuracy)

Genel olarak tüm algoritmalar incelendiğinde ortalama doğruluk oranlarının %82'nin üzerinde olduğu görülmektedir. Bu da bize başarılı bir sonuç elde etme olasılığının yüksek olduğunu gösterir. Algoritmaları değerlendirdiğimizde en yüksek doğruluk oranına sahip tahmin genellikle ev sahibi takımın kazanma tahminidir (%89-%96). Deplasman takım kazanma tahmini oranı ise yüksektir, ancak ev sahibi takım kazanma oranı kadar değildir (%86-%94). En düşük doğruluk oranı ise berabere kalma tahminindedir (%54-%57).

6.1. Kesinlik (Precision)

Algoritmaların sonuçlarına göre, ortalama kesinlik oranları %79 ile %84 arasında çıkmıştır. Bu oranlar, pozitif olarak sınıflandırılan durumların gerçekte ne kadarının pozitif olduğunu gösterir. Yani yüksek bir kesinlik oranı, modelin pozitif tahminlerinin genellikle doğru olduğunu gösterir.

6.1. Duyarlılık (Recall)

Algoritmaların elde ettiği sonuçlara göre, ortalama duyarlılık oranı %78 ile %81 arasında değişmektedir. Bu durum gerçek pozitif sonuçların etkili bir şekilde tespit etme başarısını göstermektedir ve bu algoritmanın yüksek bir duyarlılığa sahip olduğunu yansıtmaktadır, yani gerçek pozitif sonuçları kaçırma olasılığı düşüktür.

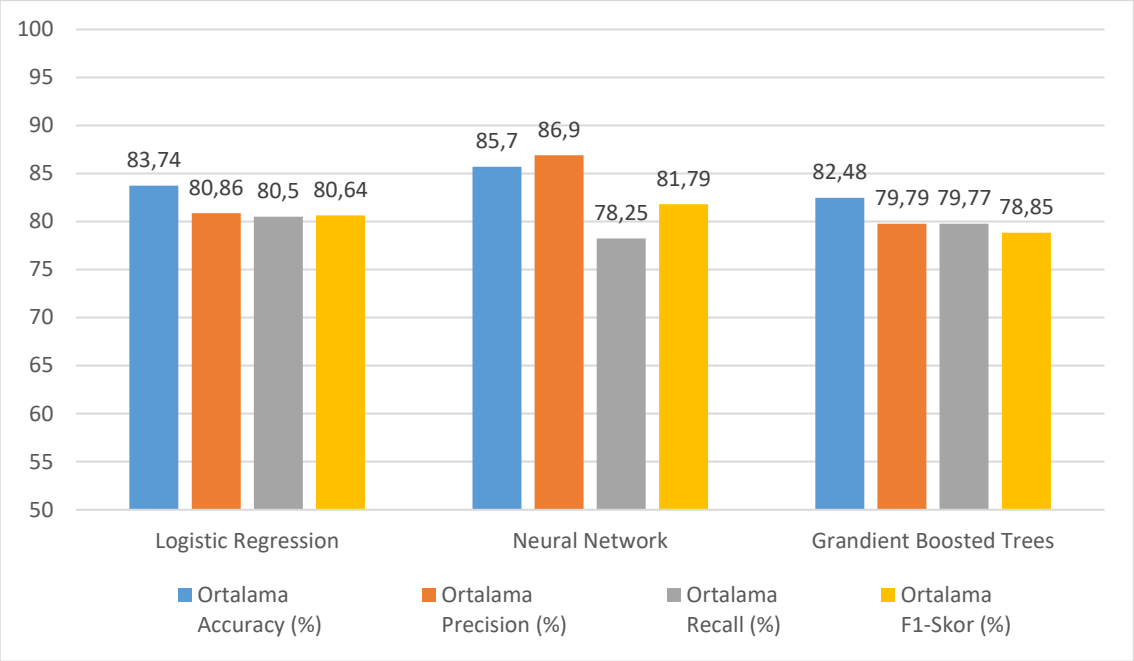
F1-Skor (F1- Measure)

Algoritmaların elde ettiği sonuçlara göre, ortalama F1-skoru oranı %78 ile %82 arasında sonuç vermiştir. Bu aralık, algoritmanın oldukça olumlu bir performans sergilediğini göstermektedir. Bu durum, algoritmanın kesinlik ve duyarlılık oranlarını dengeli bir şekilde koruyabildiğini vurgulamaktadır. Diğer bir deyişle, algoritmanın doğru pozitif sonuçları etkili bir şekilde tespit edebilmesiyle birlikte yanlış pozitif sonuçlara düşük bir eğilim gösterdiği görülmektedir.

Tablo 6.1. Kullanılan tüm algoritmalar için ortalama performans sonuçları

Performans Ölçümleri	Ortalama Accuracy (%)	Ortalama Precision (%)	Ortalama Recall (%)	Ortalama F1-Skor (%)
Logistic Regression	83.74	80.86	80.50	80.64
Neural Network	85.70	86.90	78.25	81.79
Grandient Boosted Trees	82.48	79.79	79.77	78.85

Grafik 6.2. Kullanılan tüm algoritmalar için ortalama performans sonuçlarının grafik gösterimi



Bu grafikte, her bir algoritmanın Accuracy, Precision, Recall ve F1-Skor değerleri yan yana karşılaştırılmaktadır. Grafiği incelediğinizde, en yüksek doğruluk(accuracy) değerine sahip algoritmanın "Neural Net" olduğunu, en yüksek kesinlik(precision) değerine sahip algoritmanın "Neural Net" olduğunu, en yüksek duyarlılık(recall) değerine sahip algoritmanın "Logistic Regression" olduğunu söyleyebilirsiniz. F1-Skor değerleri açısından ise en yüksek performansa yine "Neural Net " ulaştığı görünmektedir.

Logistic Regression, genel doğruluk açısından diğer algoritmalarla kıyaslandığında ortalama bir performans sergilemiştir. Kesinlik, duyarlılık ve F1-Score değerleri birbirine oldukça yakındır, bu da modelin sınıflandırma konusunda genel olarak dengeli bir performans gösterdiğini gösterir.

Neural Net, en yüksek doğruluk değerine sahiptir. Kesinlik ve f1-skor açısından da oldukça başarılıdır. Ancak, Recall değeri diğer algoritmalarla kıyaslandığında biraz düşüktür.

Gradient boosted trees, genel doğruluk açısından diğer algoritmalarla benzer bir performans gösterse de aralarında en düşük doğruluğa sahip algoritmadır. Precision, Recall ve F1-Score değerleri birbirine oldukça yakındır, bu da modelin sınıflandırma konusunda genel olarak dengeli bir performans gösterdiğini gösterir.

7. ÖNERİLER

Projemizin başlangıcında 200-300 veri ile başlayarak, çeşitli testler gerçekleştirdik ve biryandan da öğrenme sürecimizi hızlandırdık. İşin mantığını kavradıktan sonra veri setimizi 2448'e çıkardık ve genel olarak başarılı sonuçlar elde ettik. Ancak, daha fazla veri kullanarak daha sağlam ve güvenilir sonuçlar elde etmek mümkün olabilirdi. Proje sürecinde en çok kullandığımız uygulamalardan bir tanesi RapidMiner uygulaması oldu. Turbo Prep özelliği, verilerimizdeki kirli verileri temizleme ve düzeltme konusunda kolaylık sağlayarak, veri işleme adımlarını kolaylaştırmamıza büyük katkı sağladı. Ayrıca Auto Model özelliği, otomatik olarak model oluşturarak 6-7 farklı algoritma üzerinde verilerimizi test etmemize imkân tanıdı. Bu sayede hangi algoritmanın kullanılacağına dair önemli fikirler edindik.

Gelecekteki çalışmalarda daha geniş veri setleriyle çalışarak sonuçları daha da güçlendirmeyi hedefliyoruz.

8. KAYNAKLAR

[1] (Wikipedia katılımcıları, 2022, RapidMiner, https://tr.wikipedia.org/w/index.php?title=RapidMiner&oldid=28103311[Ziyaret Tarihi: 23 Aralık 2023])

[2] (Singh, J., 2023, Logistic Regression, https://www.tutorialspoint.com/difference-between-neural-network-and-logistic-regression [23 Aralık 2023])

[3] (Singh, J., 2023, Neural Network,https://www.tutorialspoint.com/difference-between-neural-network-and-logistic-regression [23 Aralık 2023])

[4] Wikipedia katılımcıları, 2023, Gradientboosting, ermanent https://en.wikipedia.org/w/index.php?title=Gradient_boosting&oldid=1190910306 [Ziyaret Tarihi: 23 Aralık 2023])

[5] (Şirin, E., 2017**,** Confusion Matrix**, https://www.veribilimiokulu.com/hata-matrisini-confusion-matrix-yorumlama/** [23 Aralık 2023**])**

[6] (Raschka, S., 2016**,** Logistic Regression and Neural Networks**, https://sebastianraschka.com/faq/docs/logisticregr-neuralnet.html** [23 Aralık 2023**])**

[7] Piryonesi, S. Madeh; El-Diraby, Tamer E. (2020). "Data Analytics in Asset Management: Cost-Effective Prediction of the Pavement Condition Index"

[8] (2022, Cross Validation, https://docs.rapidminer.com/latest/studio/operators/validation/cross_validation.html[23 Aralık 2023[)