**VERİ MADENCİLİĞİNE GİRİŞ**

**DÖNEM PROJESİ RAPORU**

Öğrenci Adı: Kaan

Öğrenci Soyadı: Sönmezöz

Öğrenci Numarası: 15011021

Veri Seti: Dota2 Games Results Data Set

**VERİ ANALİZİ**

Veri seti, Dota2 ismindeki internet üzerinden oynana bir oyunun maçlarını içermektedir. Veri setinde toplam 102.944 örnek bulunmaktadır. Veri setinde iki sınıftan oluşmaktadır. Bu sınıflar maçın hangi takım tarafından kazanıldığını belirtmektedir. Bu kategorik değerler, 1 ve -1 olarak gösterilmektedir. Verilerin 48.660 tanesi -1 ile etiketlenmişken 54.284 tanesi ise 1 ile etiketlenmiştir.

Veri setinde 116 tane özellik, 117 sütun bulunmaktadır. Herhangi bir eksik veri bulunmaktadır. Baştan sonra doğru sütunlar: maçı hangi takımın kazandığını (sınıf bilgisi), cluster id, oyun modu, oyun tipi ve geri kalan 113 sütun oyun içerisindeki karakterlerin maç esnasında hangi takımda olduklarını göstermektedir. 113 tane karakter sütunlarının bir maçta (örnekte) 10 tanesi dolu olabilmektedir. Ait oldukları takımlara göre -1 ve 1 değerlerini almaktadırlar. Eğer herhangi bir takımda değilse o karakter değeri 0 olmaktadır.

Csv formatındaki dosyayı arff formatına çevirirken sütunların isimlerine ihtiyaç duyuldu. Sitede belirtilen[1] mapping kullanılarak sütunlardaki herolar ilgili şekilde isimlendirildi. Bir tane hero için mapping eksikti. Bunun için veri setinin oluşturulması için hazırlanan kod[2] incelenerek veri çekmek için OpenDota API’nın kullanıldığı tespit edildi. OpenDota API’ya bakılarak eksik sütun bilgisi tamamlandı. Sütunların değerlerinin ne anlama geldiği ise veri seti dökümanında bulunmadığından ötürü gene veri seti için hazırlanan kod incelenerek kullanılan wrapper’ın dokümantasyonundan[3] alındı.

“game\_mode” sütunun alabileceği değerler 0’dan 22’ye kadar sırasıyla “Unknown”, “All Pick”, “Captain's Mode”, “Random Draft”, “Single Draft”, “All Random”, “Intro”, “Diretide”, “Reverse Captain's Mode”, “The Greeviling”, “Tutorial”, “Mid Only”, “Least Played”, “New Player Pool”, “Compendium Matchmaking”, “Custom”, “Captains Draft”, “Balanced Draft”, “Ability Draft “, “Event”, “All Random Death Match”, “Solo Mid 1 vs 1”, “Ranked All Pick” değerlerine karşılık gelmektedir.

“game\_type” sütunun alabileceği değerler -1’den 8’e kadar sırasıyla “Invalid”, “Public matchmaking”, “Practice”, “Tournament”, “Tutorial”, “Co-op with AI”, “Team match”, “Solo queue”, “Ranked matchmaking” ve “Solo Mid 1 vs 1” değerlerine karşılık gelmektedir.

Weka içerisinde bulunan InfoGainAttributeEval algoritması kullanılarak veri seti içerisindeki ayırt edici özellikler belirlendi. İlk beş belirleyici özellik: enchantress, mirana, life\_stealer, windrunner, abyssal\_underlord.

**SINIFLANDIRMA**

Sınıflandırma için Weka’da bulunan algoritmalar kullanıldı. Aşağıdaki tabloda kullanılan algoritmalar ve bu algoritmaların isabet oranları yer almaktadır. Bu sonuçlar verisetinin %80’i eğitim için geri kalanı test etmek için kullanıldığında elde edilmiştir. Algoritmaların varsayılan ayarları kullanılmıştır.

|  |  |
| --- | --- |
| **Sınıflayıcı** | **İsabet** |
| Logistic | %60.1438 |
| SimpleLogistic | %60.0903 |
| BayesNet | %59.8912 |
| NaiveBayes | %59.8863 |
| NaiveBayesUpdateable | %59.8863 |
| VotedPerceptron | %58.507 |
| SGD | %55.1071 |
| REPTree | %55.0002 |
| KStar | %52.2415 |
| RandomTree | %51.7267 |
| IBk | %51.4304 |

Buna göre en başarılı sonuç verenler Logistic, SimpleLogistic ve BayesNet algoritmalarıdır.

**Logistic Algoritması**

Bu algoritma Multinomial Logistic Regression modelini Ridge Estimator kullanarak yaratmaktadır[4]. Multinomial Logistic Regression; nominal output değerinin, nominal olan öznitelikleriyle arasındaki ilişkinin açıklanması gerektiğinde kullanılmaktadır[5].

|  |  |  |
| --- | --- | --- |
| n = 20589 | Predicted  -1 | Predicted  1 |
| Actual -1 | 0.2418 | 0.2364 |
| Actual 1 | 0.1620 | 0.3595 |

**SimpleLogistic Algoritması**

Logistic Algoritmasından farklı olarak LogitBoost kullanmaktadır[6].

|  |  |  |
| --- | --- | --- |
| n = 20589 | Predicted  -1 | Predicted  1 |
| Actual -1 | 0.2421 | 0.2362 |
| Actual 1 | 0.1628 | 0.3587 |

**BayesNet Algoritması**

Naive Bayes’e göre daha genel bir ilişki kurmaktadır bağımsız değişkenler ve sınıf değişkeni arasında.

|  |  |  |
| --- | --- | --- |
| n = 20589 | Predicted  -1 | Predicted  1 |
| Actual -1 | 0.2404 | 0.2378 |
| Actual 1 | 0.1631 | 0.3584 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Algoritma** | **True Positive** | **TP Rate** | **FP Rate** | **Precision** | **Recall** | **F-Measure** |
| Logistic | -1 | 0,506 | 0,311 | 0,599 | 0,506 | 0,548 |
| Logistic | 1 | 0,689 | 0,494 | 0,603 | 0,689 | 0,643 |
| SimpleLogistic | -1 | 0,506 | 0,312 | 0,598 | 0,506 | 0,548 |
| SimpleLogistic | 1 | 0,688 | 0,494 | 0,603 | 0,688 | 0,643 |
| BayesNet | -1 | 0,503 | 0,313 | 0,596 | 0,503 | 0,545 |
| BayesNet | 1 | 0,687 | 0,497 | 0,601 | 0,687 | 0,641 |

**KÜMELEME**

Weka’daki kümeleme algoritmalarından SimpleKMeans, MakeDensityBasedClusterer ve FarthestFirst algoritmaları kullanıldı. Bunların başarı oranları aşağıda yer almaktadır.

|  |  |
| --- | --- |
| **Kümeleyici** | **Yanlış Kümeleme** |
| FarthestFirst | %48.6799 |
| MakeDensityBasedClusterer | %49.3297 |
| SimpleKMeans | %49.3297 |

**KODLAMA**

**Sınıflama**

Sınıflama için Naive Bayes algoritması kodlanmıştır. Bu rapor hazırlanırken ilgili kodun çalışması devam ettiğinden ötürü herhangi bir sonuç elde edilememiştir. Aşağıda ilgili algoritmanın kodu bulunmaktadır. K-fold cross validation kullanılmıştır.

K=5 için ortalama isabet(accuracy) 0.59571 bulunmuştur. Aşağıda da confussion matrixler yer almaktadır. İçermiş oldukları değerler ortalama değerleridir.

*Confussion Matrix Normalize edilmiş Confussion Matrix*

|  |  |  |
| --- | --- | --- |
| n=102944 | Predicted  -1 | Predicted  1 |
| Actual -1 | 28946 | 19714 |
| Actual 1 | 21907 | 32377 |

|  |  |  |
| --- | --- | --- |
|  | Predicted  -1 | Predicted  1 |
| Actual -1 | 0.2811 | 0.1915 |
| Actual 1 | 0.2128 | 0.3145 |

Weka’da Naive Bayes algoritmasıyla k-fold cross validation (k = 5) kullanılarak elde edilen sonuç:

*accuracy* (isabet): % 59.7665

|  |  |  |
| --- | --- | --- |
| n=102944 | Predicted  -1 | Predicted  1 |
| Actual -1 | 24632 | 24028 |
| Actual 1 | 17390 | 36894 |

|  |  |  |
| --- | --- | --- |
|  | Predicted  -1 | Predicted  1 |
| Actual -1 | 0.2392 | 0.2334 |
| Actual 1 | 0.1689 | 0.3583 |

*Confussion Matrix Normalize edilmiş Confussion Matrix*

*Genel Karşılaştırma*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Algoritma** | **Positive Class** | **True Positive** | **FP Rate** | **Precision** | **Recall** | **F-Measure** |
| Weka  Naive Bayes | -1 | 0,506 | 0,311 | 0,599 | 0,506 | 0,548 |
| Weka  Naive Bayes | 1 | 0,689 | 0,494 | 0,603 | 0,689 | 0,643 |
| Naive Bayes | -1 | 0,506 | 0,312 | 0,598 | 0,506 | 0,548 |
| Naive Bayes | 1 | 0,688 | 0,494 | 0,603 | 0,688 | 0,643 |

|  |  |
| --- | --- |
| Algoritma | Accuracy (İsabet) |
| Weka - NaiveBayes | % 59.7665 |
| Naive Bayes | %59.571 |

**import** pandas **as** pd

**from** sklearn**.**model\_selection **import** KFold ## kendi metodunu yaz sonra vakit kalirsa

DATASET\_PATH **=** "../data/with-headers/dataset.csv"

classes **=** **{**'team\_1'**:** 1**,** 'team\_2'**:** **-**1**}**

**def** read\_dataset**(**file\_path **=** DATASET\_PATH**):**

**return** pd**.**read\_csv**(**file\_path**)**

**def** remove\_columns**(**dataset**,** columns **=** **[**"cluster\_id"**]):**

**return** dataset**.**drop**(**columns**=**columns**)**

**def** get\_class\_samples**(**dataset**):**

team\_1 **=** filter\_samples\_by**(**dataset**,** "class"**,** classes**[**'team\_1'**])**

team\_2 **=** filter\_samples\_by**(**dataset**,** "class"**,** classes**[**'team\_2'**])**

**return** team\_1**,** team\_2

**def** filter\_samples\_by**(**samples**,** column\_name**,** column\_value**):**

sample\_indexes **=** samples**[**column\_name**]** **==** column\_value

**return** samples**[**sample\_indexes**]**

**def** get\_feature\_column\_names**(**samples**):**

column\_names **=** samples**.**columns**.**values**.**tolist**()**

column\_names**.**remove**(**'class'**)**

**return** column\_names

**def** calculate\_probability\_for**(**class\_samples**,** feature\_name**,** feature\_value**,** occurences**,** class\_name**):**

sample\_count **=** class\_samples**.**shape**[**0**]**

feature\_occurences **=** occurences**[**class\_name**]**

**if** feature\_name **in** feature\_occurences**:**

feature\_occurences **=** feature\_occurences**[**feature\_name**]**

feature\_value\_string **=** str**(**feature\_value**)**

**if** feature\_value\_string **in** feature\_occurences**:**

filtered\_sample\_count **=** feature\_occurences**[**feature\_value\_string**]**

**else:**

filtered\_sample\_count **=** filter\_samples\_by**(**class\_samples**,** feature\_name**,** feature\_value**).**shape**[**0**]**

feature\_occurences**[**feature\_value\_string**]** **=** filtered\_sample\_count

**else:**

filtered\_sample\_count **=** filter\_samples\_by**(**class\_samples**,** feature\_name**,** feature\_value**).**shape**[**0**]**

occurence **=** **{**str**(**feature\_value**):** filtered\_sample\_count**}**

feature\_occurences**[**feature\_name**]** **=** occurence

**return** filtered\_sample\_count **/** sample\_count

**def** calculate\_probability\_for\_all\_features**(**class\_samples**,** test\_sample**,** occurences**,** class\_name**):**

feature\_columns **=** get\_feature\_column\_names**(**class\_samples**)**

probability **=** 1.0

**for** i **in** range**(**len**(**feature\_columns**)):**

feature **=** feature\_columns**[**i**]**

value **=** test\_sample**[**feature**]**

feature\_probability **=** calculate\_probability\_for**(**class\_samples**,** feature**,** value**,** occurences**,** class\_name**)**

probability **=** probability **\*** feature\_probability

**return** probability

**def** predict\_sample**(**class\_1\_samples**,** class\_2\_samples**,** test\_sample**,** occurences**):**

team\_1\_probability **=** calculate\_probability\_for\_all\_features**(**class\_1\_samples**,** test\_sample**,** occurences**,** "team\_1"**)**

team\_2\_probability **=** calculate\_probability\_for\_all\_features**(**class\_2\_samples**,** test\_sample**,** occurences**,** "team\_2"**)**

**if** team\_1\_probability **>** team\_2\_probability**:**

**return** classes**[**'team\_1'**]**

**else:**

**return** classes**[**'team\_2'**]**

**def** init\_prediction\_counts**():**

**return** **{**

'1'**:** **{**'true'**:** 0**,** 'false'**:** 0**},**

'-1'**:** **{**'true'**:** 0**,** 'false'**:** 0**}**

**}**

**def** predict\_all\_samples**(**train\_dataset**,** test\_dataset**):**

predictions **=** init\_prediction\_counts**()**

occurences **=** **{**

'team\_1'**:** **{},** 'team\_2'**:** **{}**

**}**

team\_1\_samples**,** team\_2\_samples **=** get\_class\_samples**(**train\_dataset**)**

test\_size **=** test\_dataset**.**shape**[**0**]**

**for** i **in** range**(**test\_size**):**

test\_sample **=** test\_dataset**.**iloc**[**i**]**

actual\_class **=** test\_sample**[**'class'**]**

predicted\_class **=** predict\_sample**(**team\_1\_samples**,** team\_2\_samples**,** test\_sample**,** occurences**)**

**print(**"Prediction: " **+** str**(**i**+**1**)** **+** " actual: " **+** str**(**actual\_class**)** **+** " predicted: " **+** str**(**predicted\_class**))**

**if** actual\_class **==** predicted\_class**:**

predictions**[**str**(**predicted\_class**)][**'true'**]** **+=** 1

**else:**

predictions**[**str**(**predicted\_class**)][**'false'**]** **+=** 1

**return** predictions

dataset **=** read\_dataset**()**

dataset **=** remove\_columns**(**dataset**,** **[**"cluster\_id"**])**

kFold **=** KFold**(**n\_splits **=** 5**,** shuffle **=** **False,** random\_state **=** **None)**

results **=** **[]**

count **=** 1

**for** train\_indices**,** test\_indices **in** kFold**.**split**(**dataset**):**

**print(**"Validation: " **+** str**(**count**))**

train\_dataset **=** dataset**.**iloc**[**train\_indices**]**

test\_dataset **=** dataset**.**iloc**[**test\_indices**]**

results**.**append**(**predict\_all\_samples**(**train\_dataset**,** test\_dataset**))**

count **=** count **+** 1

predictions **=** **{**'-1'**:** **{**'total\_true'**:** 0**,** 'total\_false'**:** 0**},**

'1'**:** **{**'total\_true'**:** 0**,** 'total\_false'**:** 0**},**

**}**

**for** i **in** range**(**len**(**results**)):**

predictions**[**'-1'**][**'total\_true'**]** **+=** results**[**i**][**'-1'**][**'true'**]**

predictions**[**'-1'**][**'total\_false'**]** **+=** results**[**i**][**'-1'**][**'false'**]**

predictions**[**'1'**][**'total\_true'**]** **+=** results**[**i**][**'1'**][**'true'**]**

predictions**[**'1'**][**'total\_false'**]** **+=** results**[**i**][**'1'**][**'false'**]**

accuracy **=** **(**predictions**[**'-1'**][**'total\_true'**]** **+** predictions**[**'1'**][**'total\_true'**])** **/** **(**dataset**.**shape**[**0**])**

**Kümeleme**

K-means algoritması kodlanmıştır. Durma noktası olarak maksimum iterasyon sayısı seçilmiştir. Kısa sürede sonuç bulabilmesi adına test ederken iterasyon sayısı küçük tutularak beş seçilmiştir. Veri setinde herhangi bir nümerik değer bulunmamaktadır. Bundan ötürü kümeleme işleminde “Jaccard Similarity” kullanılmıştır. Başlangıçta kümeler oluşturulurken kümeleri temsil eden noktalar rastgele seçilmiştir. Kümeleri temsil eden örnekler bulunurken gene “Jaccard Similarity” hesaplanır. Küme içerisindeki diğer örneklere benzerliği toplamda en fazla örnek o kümeyi temsil etmek üzere seçilir. Yeni noktalara göre tekrardan kümelere atanır noktalar. Maksimum iterasyon sayısı geçildiğinde en son oluşmuş olan kümeler hangi sınıftan örnek daha çok barındırıyorsa o sınıf ile etiketlenir. Bu yöntem Weka’da clusterer olarak “SimpleKMeans”, cluster mode olarak ise “Classes to clusters evaluation” denk düşmektedir.

**import** pandas **as** pd

DATASET\_PATH **=** "../data/with-headers/dataset.csv"

MAX\_ITERATION **=** 5

**def** read\_dataset**(**file\_path **=** DATASET\_PATH**):**

**return** pd**.**read\_csv**(**file\_path**)**

**def** remove\_columns**(**dataset**,** columns **=** **[**"cluster\_id"**]):**

**return** dataset**.**drop**(**columns**=**columns**)**

**def** get\_feature\_column\_names**(**samples**):**

column\_names **=** samples**.**columns**.**values**.**tolist**()**

column\_names**.**remove**(**'class'**)**

**return** column\_names

**def** calculate\_jaccard\_similarity**(**sample\_1**,** sample\_2**):**

common\_count **=** 0

feature\_count **=** len**(**feature\_columns**)**

**for** i **in** range**(**feature\_count**):**

feature\_name **=** feature\_columns**[**i**]**

**if** sample\_1**[**feature\_name**]** **==** sample\_2**[**feature\_name**]:**

common\_count **+=** 1

**return** **(**common\_count **/** feature\_count**)**

**def** find\_cluster**(**center\_1**,** center\_2**,** sample**):**

similarity\_1 **=** calculate\_jaccard\_similarity**(**center\_1**,** sample**)**

similarity\_2 **=** calculate\_jaccard\_similarity**(**center\_2**,** sample**)**

**if** similarity\_1 **>** similarity\_2**:**

**return** "cluster\_1"

**else:**

**return** "cluster\_2"

**def** find\_representative\_sample**(**elements**):**

**print(**"Finding representative element"**)**

total\_similarities **=** **[]**

elements\_length **=** len**(**elements**)**

**for** i **in** range**(**elements\_length**):**

**print(**"Calculating total similarity for element: " **+** str**(**i **+** 1**))**

total\_similarity **=** 0

sample **=** elements**[**i**]**

**for** j **in** range**(**elements\_length**):**

total\_similarity **+=** calculate\_jaccard\_similarity**(**sample**,** elements**[**j**])**

total\_similarities**.**append**(**total\_similarity**)**

max\_similarity **=** max**(**total\_similarities**)**

max\_value\_index **=** total\_similarities**.**index**(**max\_similarity**)**

**return** elements**[**max\_value\_index**]**

**def** assign\_new\_centers**(**clusters**):**

new\_center\_1 **=** find\_representative\_sample**(**clusters**[**'cluster\_1'**][**'elements'**])**

new\_center\_2 **=** find\_representative\_sample**(**clusters**[**'cluster\_2'**][**'elements'**])**

clusters**[**'cluster\_1'**][**'center'**]** **=** new\_center\_1

clusters**[**'cluster\_2'**][**'center'**]** **=** new\_center\_2

**def** assign\_samples\_to\_clusters**(**dataset**,** clusters**):**

center\_1 **=** clusters**[**'cluster\_1'**][**'center'**]**

center\_2 **=** clusters**[**'cluster\_2'**][**'center'**]**

**for** i **in** range**(**dataset**.**shape**[**0**]):**

**print(**"Sample " **+** str**(**i **+** 1**))**

sample **=** dataset**.**iloc**[**i**]**

cluster\_name **=** find\_cluster**(**center\_1**,** center\_2**,** sample**)**

clusters**[**cluster\_name**][**'elements'**].**append**(**sample**)**

**def** find\_cluster\_class**(**elements**):**

class\_count **=** **{**'-1'**:** 0**,** '1'**:** 0**}**

**for** i **in** range**(**len**(**elements**)):**

class\_value\_str **=** str**(**elements**[**i**][**"class"**])**

class\_count**[**class\_value\_str**]** **+=** 1

**if** class\_count**[**'1'**]** **>** class\_count**[**'-1'**]:**

**return** "team\_1"**,** class\_count**[**'1'**],** class\_count**[**'-1'**]** ## returns class\_label, true\_predictions, wrong\_predictions

**else:**

**return** "team\_2"**,** class\_count**[**'-1'**],** class\_count**[**'1'**]**

**def** clear\_cluster\_elements**(**clusters**):**

clusters**[**'cluster\_1'**][**'elements'**].**clear**()**

clusters**[**'cluster\_2'**][**'elements'**].**clear**()**

**def** init\_clusters**(**dataset**):**

center\_1 **=** dataset**.**iloc**[**0**]**

center\_2 **=** dataset**.**iloc**[**dataset**.**shape**[**0**]** **-** 1**]**

clusters **=** **{**

'cluster\_1'**:** **{**

'center'**:** center\_1**,**

'elements'**:** **[]**

**},**

'cluster\_2'**:** **{**

'center'**:** center\_2**,**

'elements'**:** **[]**

**}**

**}**

**return** clusters

dataset **=** read\_dataset**()**

dataset **=** remove\_columns**(**dataset**,** **[**"cluster\_id"**])**

feature\_columns **=** get\_feature\_column\_names**(**dataset**)**

clusters **=** init\_clusters**(**dataset**)**

**for** i **in** range**(**MAX\_ITERATION**):**

**print(**"Iteration: " **+** str**(**i **+** 1**))**

clear\_cluster\_elements**(**clusters**)**

assign\_samples\_to\_clusters**(**dataset**,** clusters**)**

assign\_new\_centers**(**clusters**)**

predictions **=** **{**'team\_1'**:** **{**'true'**:** 0**,** 'false'**:** 0**},**

'team\_2'**:** **{**'true'**:** 0**,** 'false'**:** 0**}**

**}**

class\_name**,** true\_predictions**,** false\_predictions **=** find\_cluster\_class**(**clusters**[**'cluster\_1'**][**'elements'**])**

predictions**[**class\_name**][**'true'**]** **+=** true\_predictions

predictions**[**class\_name**][**'false'**]** **+=** false\_predictions

class\_name**,** true\_predictions**,** false\_predictions **=** find\_cluster\_class**(**clusters**[**'cluster\_2'**][**'elements'**])**

predictions**[**class\_name**][**'true'**]** **+=** true\_predictions

predictions**[**class\_name**][**'false'**]** **+=** false\_predictions

**KAYNAKÇA**

[1] <https://archive.ics.uci.edu/ml/datasets/Dota2+Games+Results>

[2] <https://gist.github.com/da-steve101/1a7ae319448db431715bd75391a66e1b>

[3] <https://github.com/joshuaduffy/dota2api/blob/master/docs/source/responses.rst>

[4] <https://weka.sourceforge.io/doc.dev/weka/classifiers/functions/Logistic.html>

[5] <https://www.statisticssolutions.com/mlr/>

[6] <https://weka.sourceforge.io/doc.dev/weka/classifiers/functions/SimpleLogistic.html>