YBS Ansiklopedi

www.YBSAnsiklopedi.com

Cilt 7, Sayı 1, 2019

Dolandırıcılık Tespiti (Fraud Detection)

İzay Özaslan[1]

1. *Antalya Bilim Üniversitesi, Bilgisayar Mühendisliği Bölümü*

E-mail: izay.ozaslan@std.antalya.edu.tr

Özet

Finansal işlemlerin internet üzerinden ve mobil bankacılıkla yapılabilmesi; transferleri ve alım/satımı ne kadar kolaylaştırsa da, bir o kadar da dolandırıcılık riskini arttırmaktadır. Hileli para aktarımı yapılma oranının, mevcut mali kriz ve ekonomik durgunluk sebebiyle yükselme ihtimali vardır. Yöneticilerin dolandırıcılığı engelleyici önlemler alması gerekmektedir. Dolandırıcılık tespiti ve sahtekârlığı tahmin etme sistemi geliştirmek ve kullanmak önem taşımaktadır. Potansiyel sahtekârlık oranı, para akışındaki anormallik ve veri akışını incelemek için, veri bilimi ve veri madenciliği kullanılarak bir program geliştirilmelidir. Bu makalede para transferlerindeki dolandırıcılık tespiti üzerine geliştirilen bir çalışma açıklanacaktır. Çalışma kapsamında, kodlamasız olarak makine öğrenme algoritmalarıyla veriyi işleyip yorumlayarak görselleştirme ve raporlama sağlayan bir veri bilimi geliştirme platformu olan Knime, kodlama dili olarak SKLearn kütüphanesiyle Python ve AutoML felsefesini benimsemiş olan OptiScorer ürünleri kullanılarak karşılaştırma yapılmıştır.

Anahtar Kelimeler: *veri bilimi, veri madenciliği, dolandırıcılık tespiti, finansal işlemler*

Abstract

Although financial transactions can be done via internet and mobile banking, which also makes the selling and buying transactions easier, but it increases the fraud risk. There is a possibility that the rate of fraudulent transfers increase due to the current financial crisis and the recession. Managers need to take preventions for fraud. Developing the fraud detection and fraud prediction systems have importance. A program which is made with data science and data mining, should be developed to investigate the potential fraud rate, anomaly in the money flow and data flow. This article describes a study that is developed for fraud detection in money transfers. In the study, Knime which is a data science development platform and provides visualization and reporting by processing and interpreting data with machine learning algorithms without coding, Python programming language and the SKLearn library on the top of Python as a coding environment, OptiScorer product that adopt the AutoML philosophy are compared.

Keywords: *data science, data mining, machine learning, fraud detection, financial transaction*

1. Giriş

Günümüzde teknolojinin de ilerlemesiyle dolandırıcılık yapmak kolaylaşmış ve dolayısıyla artmıştır [1]. Finansal işlemler hızlanmış ve basitleşmiş olsa da teknolojinin bu olumlu getirisinin yanında; kredi kartı kopyalamak, transferler sırasında fazladan para aktarımı için açık bulabilmek ve bunları yaparken de gizlenebilmek gibi olumsuz etkileri de vardır. Ayrıca gün içinde milyonlarca bankacılık işlemi gerçekleşmektedir ve bunun yanında az bir miktarı sahtekârlıkla yapılmaktadır; oranlar farklı olduğu için tespit etmek de zorlaşmaktadır. Yapılan araştırma verilerinin ortalamasına göre transferlerde yapılan dolandırıcılık oranı yüzde ondan daha azdır. Kredi kartı gibi işlemlerde 0.1 ve alım/satım işlemlerinde 0.5 oranında sahtekarlık vardır [2].

Hileli para aktarımından sadece bankalar değil, müşteriler de etkilenmektedir. Eğer bankalar para kaybederse, bu eksiği kart sahiplerinden yüksek faiz oranlarıyla, yükselen kart ücretleriyle ve ekstra ücretlerle müşterilerinden çıkarmaktadır. Bu yüzden dolandırıcılık tespiti hem bankalar için hem de müşteriler için oldukça önemlidir.

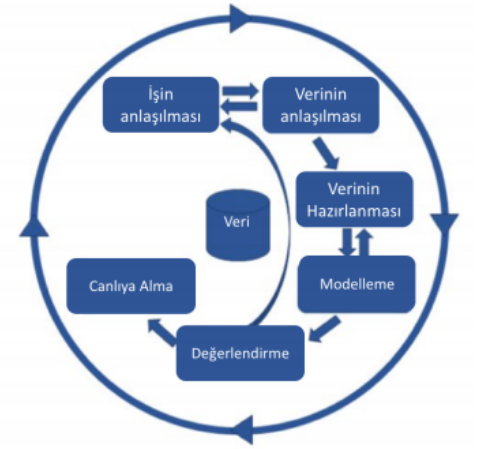
Veri bilimi ve veri madenciliği kullanarak dolandırıcılık tespiti mümkündür ve daha kolaydır. Veriler analiz edilir ve çeşitli algoritmalarla sonuçlar çıkarılıp tahmin yapılır. Veri madenciliğinde; analizi zor, büyük veri yığınlarının analiz edilip, faydalı olabilecek bilgilerin çıkarılması ve bu bilgiler sonucunda eldeki verilen sınıflandırılması veya gelecek verilerinde tahmininin yapılması amaçlanır [3].

Bu makalede öncelikle kullanılan metodolojiden bahsedilip, çalışma için kullanılan veri açıklanıp, Knime ve Python'da modelleme ve kodlama adımı anlatılıp, Optiscorer ürünleriyle karşılaştırıp, son olarak sonuçlar ve karşılaştırmalar anlatılmıştır.

2. Veri Madenciliği Aşamaları ve Yöntemleri

Bu çalışmada CRISP-DM (Cross Industry Standard Process Mode l for data Mining; Çapraz Endüstri Veri Madenciliği Standart Süreci) metodolojisi kullanılmıştır. Bu metodoloji, sorunları çözmek için kullanılan bir veri madenciliği süreci modelidir [4]. Şekil 1'de şeması gösterilmiştir. CRISP-DM veri madenciliğinde izlenen yol, 6 adımdan oluşur [5].

1) İş Süreçlerinin Anlaşılması
2) Verinin Anlaşılması
3) Veri Ön İşleme Aşaması
4) Model Aşaması
5) Değerlendirme Aşaması
6) Ürün Aşaması



Şekil 1. CRISP-DM Adımları ve Akışı

3. Veriler

Finansal hizmetler ve mobil para işlemleriyle ilgili kamuya açık veri seti paylaşımı yapılmamaktadır fakat dolandırıcılık tespiti için veri bulmak şarttır. Bu soruna çözüm olarak kaggle.com'da PaySim adı verilen yapay bir veri seti sunulmuştur. Özel veri kümelerinden toplanılan bilgilerle yasal işlemlerin yanına sahtekârlık içeren işlemler de eklenmiştir ve sunulmuştur.

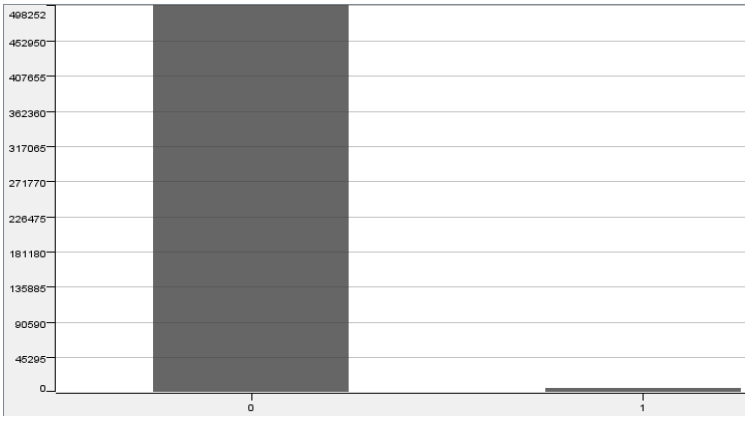
Veri setinde 11 kolon ve 2372805 satır bulunmaktadır. Veri kümesi çok büyük olduğu için küçültülerek, 498289 satır ile kullanılmıştır. Tablo 1'de veri setindeki kolonlar ve örnek veriler görülmektedir. Veri kümesi; para çekimi, para ödeme ve transfer gibi yapılan işlemleri içermektedir. Ayrıca işlem tutarı, paranın hangi müşteriden hangi müşteriye gittiği, işlem sonrası ve öncesi bakiye, işlemde belli bir miktardan fazla para aktarımı denemesi gibi bilgiler de bulunmaktadır. Kolonların açıklaması aşağıda verilmiştir.

step: Gerçek dünyada zamanın haritasını çıkarır. 1 adım 1 saate karşılık gelmektedir.
type: CASH-IN, CASH-OUT, DEBIT, PAYMENT ve TRANSFER verilerini içermektedir.
amount: İşlem tutarı verisini içermektedir.
oldbalanceOrg: İşlem öncesi ilk bakiyedir.
newBalanceOrig: İşlem sonrası yeni bakiyeyi içermektedir.
nameDest: İşlemin alıcısı olan müşterileri içermektedir.
oldbalanceDest: İşlem öncesi ilk bakiye alıcısıdır.
isFraud: İşlemin sahte olup olmadığına göre 0 (dolandırıcılık yok) ve 1 (dolandırıcılık var) verisini içermektedir.
isFlaggedFraud: Tek bir işlemde 200.000'den fazla para aktarma denemesini 0 (aktarma yok) ve 1 (aktarma var) olarak ayırmaktadır.

Tablo 1. Veri Seti

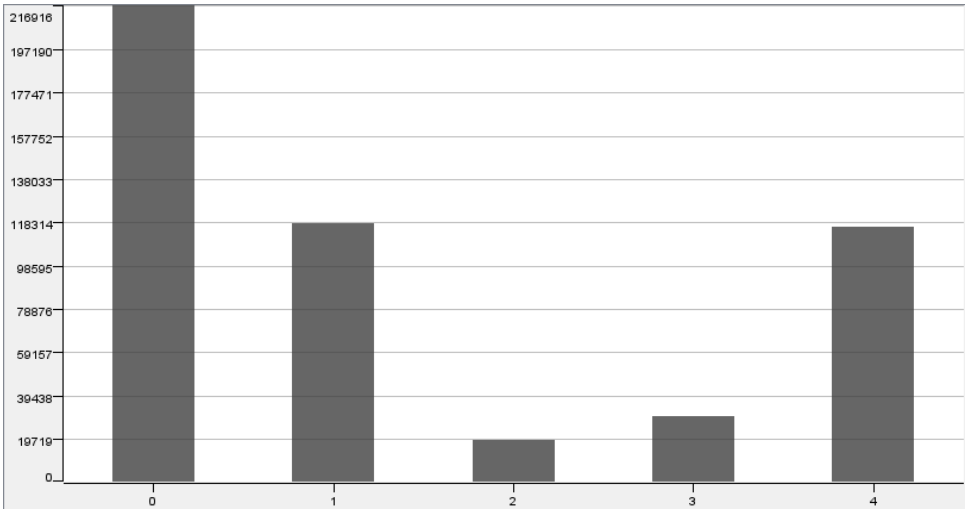
Step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
1	TRANSFER	181.0	C1305486145	181.0	0.0	C553264065	0.0	0.0	1	0
1	CASH_OUT	181.0	C840083671	181.0	0.0	C38997010	21182.0	0.0	1	0
1	PAYMENT	11668.1	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0
1	PAYMENT	7817.71	C90045638	53860.0	46042.29	M573487274	0.0	0.0	0	0
1	PAYMENT	7107.77	C154988899	183195.0	176087.23	M408069119	0.0	0.0	0	0
1	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M633326333	0.0	0.0	0	0
1	PAYMENT	4024.36	C1265012928	2671.0	0.0	M1176932104	0.0	0.0	0	0
1	DEBIT	5337.77	C712410124	41720.0	36382.23	C195600860	41898.0	40348.79	0	0
1	DEBIT	9644.94	C1900366749	4465.0	0.0	C997608398	10845.0	157982.12	0	0
1	PAYMENT	3099.97	C249177573	20771.0	17671.03	M2096539129	0.0	0.0	0	0
1	PAYMENT	2560.74	C1648232591	5070.0	2509.26	M972865270	0.0	0.0	0	0
1	PAYMENT	11633.7	C1716932897	10127.0	0.0	M801569151	0.0	0.0	0	0
1	PAYMENT	4098.78	C1026483832	503264.0	499165.22	M1635378213	0.0	0.0	0	0

Hedef kolonu olan isFlaggedFraud kolonunun histogram grafiği Grafik 1'de gösterilmiştir. Görülebileceği gibi dolandırıcılık verisi (1), dolandırıcılık olmayan veriden (0) çok daha azdır.



Grafik 1. Hedef Kolonun Histogram Grafiği

Action kolonu için oluşturulan histogram grafiği Grafik 2'de verilmiştir. Bu kolon kategorik olduğu için öncesinde nümerik veriye çevrilmiştir. Veriler; 0 (CASH_IN), 1 (CASH_OUT), 2 (DEBIT), 3 (TRANSFER), 4 (PAYMENT) olarak verilmiştir.



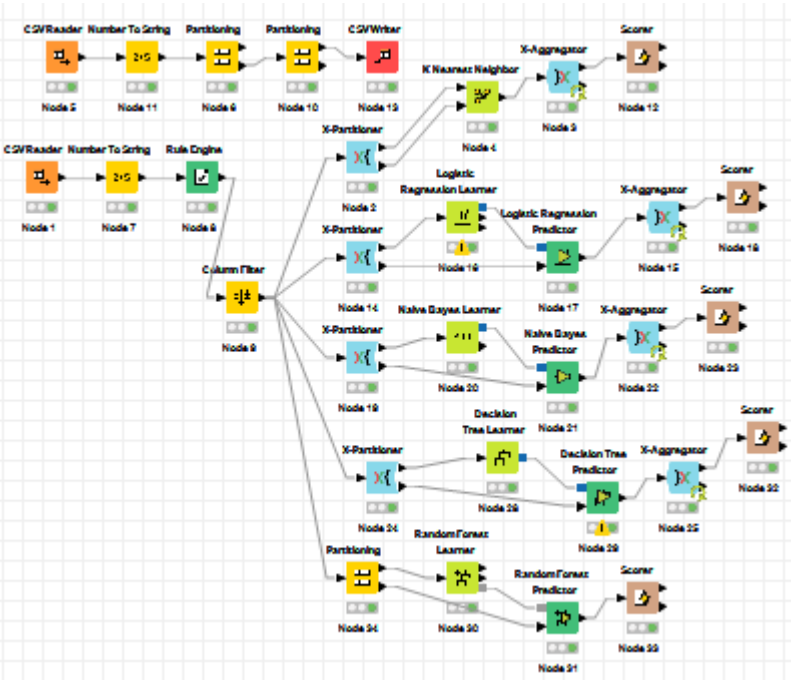
Grafik 2. Action kolonu için histogram grafiği

4. Kullanılan Programlar

Projede aynı model hem Knime hem Python kodu hem de OptiScorer üzerinden oluşturulmuş ve başarıları karşılaştırılmıştır. Knime, kodlamadan (no code) yaklaşımı ile model oluşturmakta, Python üzerinde kodlama denemeleri ile modeller test edilmiş ve AutoML yaklaşımı kullanan OptiScorer ürünü ile de aynı modeller denenmiştir. Kullanılan ürünlerin detayları bu bölümde verilecektir.

4.1 Knime

Çalışmada, veri bilimi aşamalarının modellenmesi için Knime kullanılmıştır. Knime, veri analizi ve modellemeyi düğümlerle yaparak yorumlar, analiz eder ve görselleştirir, kapsamlı bir açık kaynaklı veri entegrasyonu platformudur [6]. Çalışmanın Knime'daki modelleme görseli Şekil 2'deki gibidir.



Şekil 2. Knime'da Modelleme

Veri çok büyük olduğu için, hem Knime hem Python hem de OptiScorer'da daha hızlı işlem yapabilmek ve daha doğru sonuç alabilmek adına küçültüldü. Bunun için Knime'da Partitioning düğümü ile veri %70 oranda ayrıldı, tabakalı öğrenme (stratified sampling) seçeneği hedef kolon olan "isFraud" için seçildi ve veri eşit oranda dolandırıcılık içerdiği haliyle küçültüldü. Tabakalı öğrenme seçeneği için nümerik bir kolon seçilemeyeceğinden dolayı bölme işleminden önce hedef kolon nümerikten kategorik bir kolona dönüştürüldü. Daha sonra iki kere bölünmüş olan veri CSV Writer ile kaydedildi. Daha sonra bu veri, veri ön işleme aşamasına geçmek için CSV Reader düğümü ile yüklendi.

Number to String düğümü ile sınıflandırma için kullanılacak "isFraud" hedef kolonu, sınıflandırma algoritmalarında kullanmak üzere "integer"dan "string"e dönüştürüldü. Kategorik olan "işlem çeşidi" kolonu, encoder yapılması için Rule Engine düğümü ile kategorikten sayısal veriye dönüştürüldü. Bu işleme kadar veriler hazırlanmış oldu, bundan sonra veriyi bölme işlemine geçildi.

K-fold cross validation (k katlamalı çapraz doğrulama) kullanılarak, X-Partitioner ve Partitioning düğümü ile veri seti eğitim ve test kümesi olarak ayrıldı. K-fold cross validation, verinin önce k değeri kadar parçaya ayrılıp parçalardan birinin test, geri kalanının eğitim için kullanılması ve daha sonra her parçayı tek tek test için kullanıp geri kalanını da eğitim için kullanmaktır [7]. X-Partitioner için number of validations değerleri 10 verildi. Partitioning düğümünde %66 oranla test ve eğitim kümesi bölündü ve hedef kolon eşit oranda dağıtıldı. Random Forest algoritmasında X-Partitioner düğümü hafıza (memory) hatasına sebep olduğu için Partitioning düğümü kullanıldı. Diğer algoritmalar için X-Partitioner düğümü kullanıldı. Logistic Regression, Naive Bayes, Decision Tree, Random Forest ve K Nearst Neighbor algoritmaları için gerekli düğümler eklendi.

Logistic Regression, sınıflandırma problemlerinde kullanılan regresyon analizidir. Regresyon analizi, bir bağımlı değişken ile bir veya daha fazla nümerik değişkenin arasındaki ilişkiyi yorumlamak için kullanılır [8].

Naive Bayes teoremi, sınıf değişkenlerinin değeri verilen her bir özellik çifti arasında koşullu bağımsızlık varsayımıyla uygulanmasına dayanır [9].

Decision Tree algoritmasında veriyi alt gruplara bölerek istenen sınıfları izole etmektir [10].

Random Forest içinde birçok karar ağacı bulunduran bir sınıflandırma algoritmasıdır [11].

KNN algoritması, eğitim verisini kullanarak k değerine göre test verisini sınıflandırır [12].

Son adımda Scorer düğümü eklenerek başarılarına bakıldı. Hedef kolonumuzda, dolandırıcılık olması oranı, olmaması oranına göre çok az olduğu için başarı her zaman çok yüksek çıktı. Bu yüzden başarıyı görmek için diğer değerlere de bakıldı ve Tablo 3'te gösterildi. En başarılı algoritmanın Random Forest olduğu görüldü. Tablo 4'te confusion matrix değerleri de gösterildi.

Tablo 3. Başarı Değerleri

	Accuracy	Cohen's Kappa	F-measure
Logistic Regression	99,873 %	0,062	0,062
Naive Bayes	99,993 %	0	-
Decision Tree	99,998 %	0,824	0,824
Random Forest	99,999 %	0,917	0,917
KNN	99,997 %	0,767	0,767

Tablo 4. Confusion Matrix Sonuçları

Logistic Regression

Row ID	0	1
0	497634	618
1	16	21

Naive Bayes

Row ID	0	1
0	498252	0
1	37	0

Decision Tree

Row ID	0	1
0	498249	3
1	9	28

Random Forest

Row ID	0	1
0	169406	0
1	2	11

KNN

Row ID	0	1
0	498252	0
1	14	23

Bu çalışmadaki gibi dengesiz veriler için smote veya undersampling yöntemi kullanılabilir. Smote yöntemi, sentetik olarak azınlık gruptan örnekleme yapmaktır [13]. Mantıklı örnekleme yapılabilmesi için en yakın komşuya

bakılarak tahmin yapılır. Undersampling ise eldeki fazla veriyi azaltmak ve dengeli bir şekilde verilerin örneğini almaktır [14].

4.2 Python Kodlaması

```
#1. kutuphaneler
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

#2.1. Veri Yukleme
data = pd.read_csv('train_son.csv')
#data frame dilimleme
x = data.iloc[:,[1,2,3,5,6,8,9,10]] #bağımsız değişkenler
y = data.iloc[:,11].values #bağımlı değişkenler
names = {0:'Not Fraud', 1:'Fraud'}
print(data.isFraud.value_counts().rename(index = names))

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
x['action'] = le.fit_transform(x['action'])
print(x)
```

Kod 1. Veri Ön İşleme

Kod 1'de gösterildiği gibi öncelikle gerekli olabilecek kütüphaneler yüklenmiştir ve veri ön işleme aşamasına geçilmiştir. Daha sonra veri seti de eklenmiştir, Knime'da küçültülen veri seti kullanılmıştır. Veriler bağımlı ve bağımsız değişkenler olarak ikiye ayrılmıştır. Veri setimizde 498252 tane not fraud, 37 tane fraud verisi olduğu görülmüştür. Label encoder kullanarak, başarıyı düşürmemesi için kategorik olan kolon sayısal bir kolona dönüştürülmüştür (0,1 gibi değerler verilmiştir).

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=0)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
```

Kod 2. Verinin Eğitim ve Test Olarak Bölünmesi

Kod 2'de gösterildiği gibi veri seti eğitim ve test olarak %66 oranda bölünmüştür ve daha sonra Standard Scaler ile ölçekleme yapılmıştır. "x_train"den öğrenilip "X_train"e dönüştürme yapılmıştır. "X_test" için yeniden öğrenme yapılmayıp, öğrenilmiş olan kullanılmıştır.

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators = 10, criterion = "entropy")
rfc.fit(X_train,y_train)
y_pred = rfc.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print('RFC')
print(cm)
print('report', classification_report(y_test, y_pred))
```

Kod 3. Random Forest Algoritması

Knime'da en başarılı sonucu Random Forest Algoritması verdiği için Kod 3'teki görüldüğü üzere Python'da bu algoritmanın kodlaması yapılmıştır. Confusion matrix çıktısını çaprazlama yaparak topladığımızda doğru ve yanlış verileri hesaplamamız mümkündür.

```
from sklearn.model_selection import cross_val_score
basari = cross_val_score(estimator = rfc, X=X_train, y=y_train, cv=4)
print(basari.mean())
print(basari.std())

from sklearn.model_selection import GridSearchCV
p = [{'n_estimators':[1,2,3,4,5,6,7,8,9,10], 'criterion':['entropy']}]

gs = GridSearchCV(estimator= rfc,
                  param_grid= p,
                  scoring= 'accuracy',
                  cv= 10,
                  n_jobs= -1)
grid_search= gs.fit(X_train, y_train)
eniyisonuc = grid_search.best_score_
eniyiparametreler = grid_search.best_params_
print(eniyisonuc)
print(eniyiparametreler)
```

Kod 4. Başarı Sonuçları

Kod 4'te, başarı ve standard deviation sonuçlarını veren k fold corss validation kodu gösterilmiştir. Bunun yanında precision, recall, f1-score ve support sonuçlarını rapor eden Grid Search kodlarını içermektedir. En iyi sonucu veren parametreler ve bunun sonucundaki değere bakılmıştır.

Tablo 5. Sonuçlar

```
[[164422      0]
 [     4     10]]
report              precision    recall  f1-score   support

           0       1.00      1.00      1.00    164422
           1       1.00      0.71      0.83        14

   micro avg       1.00      1.00      1.00    164436
   macro avg       1.00      0.86      0.92    164436
weighted avg       1.00      1.00      1.00    164436

0.9999790327328968
1.3056257605950316e-05
0.9999790326880393
{'criterion': 'entropy', 'n_estimators': 5}
```

Python sonuçları Tablo 5'teki gibidir. Confusion matrix sonuçları ilk sırada görülmektedir Devamında sırasıyla rapor sonuçları, başarı (accucary), standart sapma (standard deviation), en yüksek değer ve bu değeri veren parametreler de görülmektedir.

Başarı değerlendirmesi için aşağıdaki formüller kullanılır ve bu çalışmadaki sonuçları da gösterilmiştir.

Sensitivity: $\frac{TP}{P} = \frac{TP}{TP+FN}$ 1

Specificity: $\frac{TN}{N} = \frac{TN}{TN+FP}$ 0,714

Precision: $\frac{TP}{TP+FP}$ 0,99

Accuracy: $\frac{TP+TN}{Total}$ 0,99

F1-Score: $\frac{2TP}{2TP+FP+FN}$ 0,99

Başarı değerlendirmesindeki formüller için kısaltmalar aşağıda verilmiştir.

Gerçek Pozitif (True Positive): GP (TP)

Gerçek Negatif (True Negative): GN (TN)

Yanlış Pozitif (False Positive): YP (FP)

Yanlış Negative (False Negative): YN (FN)

4.3 Optiscorer

OptiScorer, içerdiği skorlama ve sıralama algoritmalarıyla dijitalleşebilen bütün varlıklar üzerinde çalışabilmektedir. Çalışma alanları müşteri skorlama, müşteri kaybı tahmini, dolandırıcılık riski skorlama, borç tahsilatı skorlama, çalışan performansı skorlama ve risk skorlamadır [15]. Bu çalışmada OptisScorer algoritmaları ile aynı veri kümesi skorlanmıştır. Threshold (eşik) değeri 0,999 olarak hesaplanmış ve bu değere göre Tablo 6'daki sonuçlar ortaya çıkmıştır.

	Pozitif	Negatif
Pozitif	TP = 208026	FN = 5511
Negatif	TN = 0	TP = 16

Tablo 6. OptiScore Confusion Matrix Sonuçları

Knime, Python ve Optiscorer sonuçlarının karşılaştırması Tablo 7'de verilmiştir.

	Knime	Python	OptiScorer
Accuracy (başarı):	99 %	99 %	97 %
Specifity:	0,846	0,714	1,0
Sensitivity (hassasiyet):	1,0	1,0	0,97
F1:	0,99	0,99	0,98
Precision:	0,99	0,99	1,0

Tablo 7. Karşılaştırma

5. Sonuç

Bu çalışma, finansal işlemlerdeki dolandırıcılık tespiti üzerine hazırlanmış verilerden elde edilen, sınıflandırma veri yapısına sahip bir veri madenciliği projesidir. Kaggle.com'dan hazır bir seti alınmış, Knime platformu üzerinde çeşitli algoritmalar (Decision Tree, Random Forest ,Naive Bayes, K Nearest Neighbor, Logistic Regression) modellenerek karşılaştırılmış ve en başarılı sonucu veren algoritma olan Random Forest algoritmasının Python ile kodlaması yapılmıştır. Daha sonra OptiScorer ürünü sonuçlarına da bakılmıştır.

Değerlendirme için başarı (accuracy) değerine bakmak doğru sonucu vermeyecektir, çünkü veri setinde dolandırıcılık yapma oranı yapmama oranına göre çok düşük olduğu için dolandırıcılık tahmin edilemese bile doğruluk değeri %99 çıkmaktadır. Bu sebepten dolayı specifity skoru baz alınarak değerlendirme yapılmıştır.

OptiScorer ürünü, Knime ve Python sonuçları oldukça yakındır. Fakat Knime ve Python'a kıyasla OptiScorer ürünü dolandırıcılık olan bütün transferleri tespit edebilmiştir, Random Forest algoritması hepsini tespit edememiştir. Bu yüzden tespitlerde OptiScorer'ı kullanmak daha başarılı olacaktır. Bu tespit sayesinde OptiScorer ürününü kullanan firmalar dolandırıcılığı eksiksiz bir şekilde tespit edip, ileride yapılabilecek olan dolandırıcılık suçunu ön görebileceklerdir.

Kaynakça

[1] Barker, K., D'Amato, J., Sheridon, P.; (2008), "Credit card fraud: awareness and prevention", *Journal of Financial Crime*, Vol. 15 No. 4, pp. 398-410.

[2] Phua, C., Lee, V., Smith, K., Gayler, R.; (2010), "A Comprehensive Survey of Data Mining-based Fraud Detection Research", arxiv: 1009.6119, 2010

[3] Coşkun, C., Baykal, A.; (2011) "Veri Madenciliğinde Sınıflandırma Algoritmalarının Bir Örnek Üzerinde Karşılaştırılması", *Akademik Bilişim,* Şubat, 2011

[4] Rocha, B., Junior, R.; (2010), "Identifying Bank Frauds Using CRISP-DM and Decision Trees", International Journal of computer science Informaiton Technology, 2(5), 162-169, 2010

[5] Şeker, Ş.E.; (2018) , "CRISP-DM: Endüstriler Arası Standat İşleme – Veri Madenciliği İçin (Cross Industry Standard Processing – Data Mining*)", YBS Ansiklopedi*, c. 5, s.2, pp. 10-16, Temmuz, 2018

[6] Jagla, B., Wisveled B., Coppee, J.; (2011), "Extending KNIME for Next-Generation Sequencing Data Analysis", 27(20):2907-2099, October, 2011

[7] Refaeilzadeh, P., Tang, L., Liu, H.; (2008), "Cross-Validation", *Arizona State Univercity,* pp. 1-4, 2008

[8] Korkmaz, M., Güney, S., Yiğiter, Ş.; (2012), "The importance of logistic regression implementations in the Turkish livestock sector and logistic regression implementations/fields", 16(2): 25-36

[9] « 1.9 Naive Bayes » , 2007 [Çevrimiçi]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html. [Erişildi: Eylül 2019]

[10] « Section 6.3 Decision Tree Family » , [Çevrimiçi]. Available: https://www.knime.com/knime-introductory-course/chapter6/section3. [Erişildi: Eylül 2019]

[11] Goel, E., Aabhilasha E.; (2017), "Random Forest: A Review", pp. 251-257

[12] Kataria, A., Singh, M. D.; (2013), "A Review of Data Classification using K-Nearest Neighbour Algorithm", *Int. J. Emerg. Technol. Adv. Eng.,* 3(6), 354-360, June, 2013

[13] Wang, J., Xu, M., Wang, H., Zhang, J.; (2006), " Classification of Imbalanced Data by Using the SMOTE Algorithm and Locally Linear Embedding", Vol. 3, pp. 16-20, 2006

[14] Liu, X., Wu, J., Zhou, Z.; (2009), "Exploratory Undersampling for Class-Imbalance Learning", *IEEE Trans. Syst.Man Cybern Part B,* 39(2), 539-550, 2009

[15] [Çevrimiçi]. Available: https://www.optiscorer.com/. [Erişildi: Eylül 2019]