T.C.

İSTANBUL AYDIN ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

MAKİNE ÖĞRENMESİ İLE SİGORTA SEKTÖRÜNDE SAHTE HASAR TESPİTİ

YÜKSEK LİSANS TEZİ

Yaşar Geren

Bilgisayar Mühendisliği Ana Bilim Dalı

Bilgisayar Mühendisliği Programı

EKİM, 2020

T.C.

İSTANBUL AYDIN ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

MAKİNE ÖĞRENMESİ İLE SİGORTA SEKTÖRÜNDE SAHTE HASAR TESPİTİ

YÜKSEK LİSANS TEZİ

Yaşar GEREN

(Y1713.010080)

Bilgisayar Mühendisliği Ana Bilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: DR. ÖĞRETİM ÜYESİ AHMET GÜRHANLI

EKİM, 2020

YEMİN METNİ

Yüksek Lisans tezi olarak sunduğum "MAKİNE ÖĞRENMESİ İLE SİGORTA SEKTÖRÜNDE SAHTE HASAR TESPİTİ" adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya'da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (20/10/2020)

Yaşar Geren

ÖNSÖZ

İlk olarak tez konusunu belirlememde bana yardımcı olan, planlanmasında, araştırılmasında ve geliştirilmesi esnasında desteğini esirgemeyen, tecrübe ve bilgilerinden yararlandığım sayın tez danışmanım Dr. Öğr. Üyesi Ahmet GÜRHANLI'ya teşekkürü bir borç bilirim. Ayrıca bu tez çalışmam esnasında özellikle makine öğrenmesi ile ilgili teknolojiler konusunda tecrübelerini paylaşan sevgili dostum Levent Serinol' a teşekkürlerimi sunarım. Beni bu günlere getiren her türlü fedakarlığı esirgemeyen aileme ve her zaman her konuda sevgisiyle bana kuvvet veren, beni her daim destekleyen, her zaman yanımda duran sevgili eşime ve kızıma sonsuz teşekkür ediyorum.

Ekim 2020 **Yaşar Geren**

İÇİNDEKİLER

Sayfa

KISALTMALAR

ML:	Makine Öğrenmesi
AI:	Yapay Zeka
RF:	Rasgele Orman
LN:	Doğrusal Regresyon
LR:	Lojistik Regresyon
KNN:	K En Yakın Komşular
DT:	Karar Ağaçları
SVM:	Destek Vektör Makineleri
NB:	Naive Bayes
IT:	Bilgi Teknolojileri
k-CV:	K Parçalı Çapraz Doğrulama
TSB:	Türkiye Sigorta Birliği
SİSBİS:	Sigorta Suistimalleri Bilgi Paylaşım Merkezi
SBM:	Sigorta Birliği Merkezi
ABI:	İngiliz Sigortacılar Birliği
NICB:	Amerikan Uluslararası Sigortacılık Suç Bürosu

TABLO LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

MAKİNE ÖĞRENMESİ İLE SİGORTA SEKTÖRÜNDE SAHTE HASAR TESPİTİ

ÖZET

Sigorta sektöründe dolandırıcılık dünya genelinde ciddi artış göstermektedir. İnternetin günlük hayatın yanında iş hayatında ciddi oranda kullanılması ve banka kartı, kredi kartı gibi dijital ödeme araçlarının çokça kullanımıyla dolandırıcılık ve sahte işlem vakalarının artış gösterdiği görülmektedir. Özellikle finans sektörü krediler ve kredi kartları dolandırıcılıklarının tespit edilmesi ve önlenmesine ilişkin bir çok çalışma yapıldığı ve önlemler alındığı görülmektedir. Benzer durum sigorta hasarlarında yani sigortacılık sektöründe yoğun bir şekilde görülmektedir. Sigortacılıkta kullanılan sahte hasar yöntemlerinin tahmini ile ilgili uygulamaların finans alanında yapılan çalışmalara nazaran daha az olduğunu ve yapılmış çalışmalar incelendiğinde daha zor uygulamalar olduğu görülmektedir. Teknolojinin hızla ilerlemesi ve iş hayatında bu teknolojilerden faydalanılması yapılan işlemlerin hacimlerini ve dolayısıyla verileri çok hızlı bir şekilde arttırmaktadır. Yazılım dünyasında artık algoritmaların programlar tarafından otomatik yazılmasından, öğrenmesinden ve bunları uygulamasından bahsedilmektedir. Makine öğrenmesi olarak isimlendirilen bu yöntemler yapay zekanın bir alt uygulamasıdır ve veri üzerinden öğrendiğini deneyip, geliştirip bize sonuçları verir. Makine öğrenmesi yöntemleri iş hayatında bir çok sektörde kullanıldığı gibi finans ve sigorta sektöründe de yoğun bir şekilde kullanılmaktadır. Sigortalıların verileri ve hasar bilgileri büyük veri havuzunda toplanıp bu veri üzerinden ciddi analizler yapılmaktadır. Bu verilerin doğru yöntemler ve makine öğrenimi algoritmaları ile analizleri yapılarak özellikle sahte hasarların yüksek oranlarda tahmin edilmesini sağlamaktadır. Tezimde özel bir sigorta şirketinden temin edilen gerçek hasar veri seti ile makine öğrenmesi algoritmalarını kullanarak, sahte hasarların tahmin edilme skorları karşılaştırılmıştır. 7 adet değişik algoritma aynı veri seti ile eşit test ve eğitim oranları ile çalıştırılıp, doğruluk oranları ve performans değerleri karşılaştırılmış ve sonuçlar grafikler ve tablolar yardımıyla gösterilmiştir.

Anahtar Kelimeler: *Makine Öğrenimi; Makine Öğrenmesi Algoritmaları; Sahte Hasar; Büyük Veri; Sahte Hasar Tespiti;*

FRAUD DETECTION IN THE INSURANCE SECTOR WITH MACHINE LEARNING

ABSTRACT

Fraud in the insurance industry is rising on the worldwide. With the increase use of the Internet in daily life as well as in business life and increasing use of credit cards, fraud and forgery cases are also increasing. Especially in the financial sector, it is observed that many studies have been carried out and measures have been taken to detect and prevent loans and credit card frauds. A similar situation is observed intensely in insurance claims in the insurance sector. It is seen that fake claims methods used in insurance are more difficult to predict than in the financial sector and the number of studies in this area is less. The rapid advancement of technology and the use of these technologies in business life increases the volume of transactions and therefore the data very quickly increasing. In the software world, it is now mentioned that algorithms are automatically written by programs, learning and applying them. These methods, called machine learning, are a sub-application of artificial intelligence and they try and develop what they learn over data and give us results. Machine learning methods are used extensively in the finance and insurance sector as well as in many sectors in business life. The data and damage information of the insured are collected in a large data pool and serious analyzes are made on this data. By analyzing these data with correct methods and machine learning algorithms, it ensures that false damages are predicted at high rates. In my thesis, the real damage data set obtained from a private insurance company was compared with the prediction scores of fake claims by using machine learning algorithms. 7 different algorithms are run with the same data set with equal test and training rates, their accuracy rates and performance values are compared and the results are shown with the help of graphs and tables.

Keywords: *Machine Learning; Machine Learning Algorithms; Fraud Detection; Fake Claim; BigData;*

1. GİRİŞ

Sigorta ödenen bir prim karşılığı öngörülmeyen ve ileride oluşabilecek her türlü zarara karşı alınan bir önlem ve sigorta şirketi ile yapılan bir sözleşme olarak tarif edilebilir. (Demirci, 2019). Sigorta sektöründe sahte hasar bildirimleri sigortalıların primleri ile oluşturulan fonları tüketmekte ve sisteme ciddi zararlar vermektedir. Böylece gerçekten oluşan hasarların zararlarının sigorta şirketleri tarafından karşılanması zorlaşmaktadır. Sigorta hasarlarında gerçek dışı bildirimler yani suistimaller, diğer sigortalıların mağduriyetine sebebiyet veren ve önemle incelenmesi gereken ciddi bir suç ve kabahattir (Yıldırım, 2013). Dürüst sigortalı insanların ödediği primlerin büyük bir kısmı yüksek bedellerle bildirilen sahte hasarları ödemek için kullanılmakta ve bu ciddi sorunu sigorta sektörü çözmek için her tür önlemi almaya çalışmakta ve yatırımı yapmaktadır.

Sigorta sektöründe bu önlemleri almak için makine öğrenmesiyle geliştirilen uygulamaların varlığı ve sayılarının her geçen gün arttığı gözlenmektedir. Makine öğrenmesi veri setlerinde belirli bir yapı ve kalıp ortaya çıkarıp ve bu kalıplar ile bilinmeyen sonuçlara istinaden tahminlerde bulunur. Kalıp, örüntü çıkarma, makine öğrenmesi algoritmaları kullanılarak verilerin düzenleri bulunmaya çalışılır ve bu düzenlilik ile veriler değişik kategorilerde sınıflandırılır (Bishop, 2006). Makine öğrenmesini genel olarak denetimli ve denetimsiz olmak üzere iki yönteme ayırırız. Denetimli model olarak regresyon ve sınıflandırma algoritmaları sayılabilir. Denetimsiz modelleri ise kümeleme algoritmaları daha çok oluşturmaktadır (Kızılkaya ve Oğuzlar, 2018).

Sigorta hasarları ile oluşturulan veri setleri daha çok belirleyici verilerden ve alanlardan oluştuğu için ve tahminler bu belirleyici veriler üzerinden gerçekleştiği için denetimli öğrenme modellerinin sıkça kullanıldığını görmekteyiz (Günbatar, 2019). Türkiyede sigorta sektörüne özel sahte hasarların tespiti için kullanılan ticari ürünlerin kullanımının görüldüğü gibi sigorta şirketlerinde teknoloji birimleri içinde kurulan büyük veri takımlarının sayıları her geçen gün artmakta ve bu ekiplerin geliştirdiği

uygulamalar ile sahte hasarların tespit edilmesi ve öngörülmesi ile ilgili çalışmalar yapılmaktadır (www.turkishtimedergi.com, 2016), (www.sas.com, 2018).

1.1 Tezin Amacı

Tezimin amacı, sigorta sektöründe oto hasarlarında sahte olan bildirimleri makine öğrenmesi algoritmaları ile tahmin edip bu algoritmların doğruluk oranlarını ve performanslarını karşılaştırmaktır. Algoritmalar başarılı veri sayısı üzerinden doğru tahmin oranları ve hesaplama sürelerine göre karşılaştırılmıştır ve en iyi değerlere sahip algoritmalar belirlenmiştir. Uygulamamızda sahte hasarların ne oranda doğru tespitinin yapıldığı ölçülmüştür ve bu işlem veri setimiz üzerinde bulunan sahte hasar bildirimleri ile öğretime sokulup akabinde test verimiz üzerinden de tahmin gerçekleştirilerek yapılmıştır. Hesap süreleri algoritmaların toplam işlem sürelerini ölçerek elde edilmiş birbirleri arasında hızlarına göre karşılaştırma yapmak amacıyla kullanılmıştır. Uygulamamda özel bir sigorta kurumundan alınan ve gerçek hasar kayıtları ile oluşturulmuş veri seti kullanılmıştır. Veri seti sigorta şirketi tarafından kullanılan ve gerçek verileri içeren hasar departmanının sahte hasarları tespiti için hazırlanmış veri setidir. 3000 adet hasar kaydından oluşan ve rasgele seçilmiş veri setimizde hasar ekibinin tanımladığı hazır kural setleri bulunmaktadır.

1.2 Literatür Araştırması

Literatür araştırması yapıldığında bu konu ile ilgili yapılan çalışmaların daha çok finans ve bankacılık sektöründe özellikle kredi kartları sahtecilik işlemlerinin tespiti ve sigorta sektöründe de sağlık sigortaları için sahtecilik işlemlerinin tespit edilmesi ile ilgili yapılan çalışmalar olduğu görülmektedir. Elementer branş olarak geçen motorlu taşıt sigorta hasarları için sahte bildirimlerin tespit edilmesine yönelik çok örnek bulunmamaktadır. Sigorta sektörüne dair yayınlar ve şirketlerin kendi bildirimleri incelendiğinde insuretech olarak geçen yeni nesil teknolojik çalışmaların ve inovasyonların kullanıldığını ve bu teknolojilerin getirdiği maliyet avantajlarından faydalanılmaya çalışıldığı gözlenmektedir. Global ürünlere bakıldığında sigorta alanında yoğun olarak Friss fraud detection, SAS Software, NS8, Ravelin, Scorto InsuSafe ve Araxxe isimli yazılımların yoğun kullanıldığı göze çarpmaktadır.

Akademik alanda yapılmış olan örnek tezler ve çalışmaları incelediğimizde belli başlı bazı çalışmalar altta verilmiştir. Hazım (2018) 507937 sayılı tez çalışmasında kredi kartları için dolandırıcılık işlemlerinin tespit edilmesi amacıyla 4 sınıflandırma algoritmasını test etmiştir (Support Vector Machines, Naive Bayes, K Nearest Neighbors, Random Forest). Çalışmasında sınıfların yüksek dengesizliği nedeniyle alt örneklemeleri kullanmış ve modellerinin doğruluk oranlarını ölçümlemek için k parçalı çapraz validasyon yöntemini kullanmış ve bu dört algoritmayı test etmiştir. Sonuçlarda rasgele orman algoritmasının diğer algoritmalara göre daha iyi sonuçlar verdiği ve daha yüksek performans gösterdiğini belirtmiştir. Aşuk (2010) 258531 numaralı tez çalışmasında sağlık sigortaları için veri madenciliği ile sigorta susitimalleri tespiti üzerinde incelemeler yapmıştır. Bu çalışmada gaussian kernel ve linear kernel fonksiyonları modellerinin incelenmesine yer vermiştir. Çalışma sonucunda tiplerine istinaden üç bölüme ayırdığı suistimalleri linear kernel ve Gaussian kernel fonksiyonlarından hangi fonksiyonun hangi tipte suistimal için daha başarılı olduğunu belirlemeye çalışmıştır. Jagdish Pathak (Accounting Systems and IT Auditing Area, Odette School of Business Administration, University of Windsor, Windsor, Ontario, Canada) 2005 te fuzzy logic math kombinasyonları ile sigorta sektöründe hasar bildirimleri içinde sahte olan bildirimleri tespit etmek için bir çalışma yapmıştır. Kanada' da yürürlükte olan gizlilik hükümleri nedeniyle gerçek veriler kullanılamamıştır fakat test edilen fuzzy logic kombinasyonları için sigorta sahte hasar bildirimlerinin tespit edilmesinde iyi sonuçlar elde edildiğini belirtmiştir. Günbatar (2019) 587949 sayılı tez çalışmasında oto sigortaları sahteciliklerinin makine öğrenimi ile tespitine dair bir çalışma yapmıştır. Çalışmasında karar ağaçları, lojistik regresyon, yapay sinir ağları, naive bayes ağlarını ve destek vektör makinelerini test etmiştir ve sahte hasar tespitinde bütün algoritmalar için yüksek performanslar elde ettiğini, hiçbirinin diğerine çok büyük farklar göstermediği sonucuna varmıştır. Karamanlı (2019) 579835 numaralı tezinde makine öğrenimi algoritmaları ile metin madenciliği ve duygu analizi yaparak müşterilerin deneyimlerinin geliştirilmesi konusunu incelemiştir. Çalışma sonucu pozitif ve negatif yorumların tahmin edilmesiyle birlikte dikkate alınarak en iyi sonucların Destek Vektör Makineleri algoritmasıyla elde edildiğini belirtmiştir. Aleskeov, Freisleben ve Rao (1997) gerçekleştirdikleri çalışmalarında kredi kartları sahteciliği için sinir ağları öğrenmesi yöntemine dayalı veri tabanı madenciliğini kullanmışlar ve sinir ağları öğrenimi modeliyle elde ettikleri sonucun gayet başarılı tespit oranlarında olduğunu belirtmişlerdir.

1.3 Sigortacılık ve Sigorta Sektöründe Sahtecilik Kavramı

Sigorta kelimesi, kökü Latinceden gelen güven anlamındaki sicurta kelimesinden Türkçemize geçmiştir. Sigorta, sigortacının bir prim karşılığı herhangi birinin maddi değeri olan ve yasalarla korunmaya değer bir menfaatine hasar, zarar verecek bir riskin doğması durumunda bu maddi zararı karşılayabilecek tutarlarda sigortalı kişi veya şirketin tazminat almasına hak kazanımını sağlayan karşılıklı düzenlenen bir sözleşmedir. Düzenlenen sigorta poliçesinin hukuki açıdan geçerli olabilmesi için ilgili tarafların sigorta kapsamına alınan riskin, sigorta bedelinin, süresinin, şartlarının, sigortanın konusunun ve primin üzerinde mutabık kalınması gerekmektedir. Sigortacılığın amacı, hasarları önceden öngörüp önlemek değil, oluşan hasarın bütün sigortalılar arasında bölüştürerek, hasar sonucunda meydana gelen mali yükü sigortalıların her biri için katlanılabilir olmasını sağlamaktır. (Güvel ve Güvel, 2008) Ekonomik malların, ürünlerin korunması amacıyla başlayan sigortacılık için iktisadi bir dayanışma aracı olduğunu ve bu şekilde geliştiğini söyleyebiliriz. (Özbolat, 2009)

Türkiye' de sigortacılık faaliyetleri ilk olarak 1872 yılında İngiliz şirketilerinin açtıkları temsilcilikler aracılığıyla başlamıştır. İngiliz şirketlerinden sonra Fransızlar da ilgi göstermiş ve 1878 yılında ilk Fransız şirketi Türkiye' de faaliyet göstermeye başlamıştır. Daha sonra İsviçreli, İtalyan ve Alman sigorta şirketlerinin de çalışmaları ile sigortacılık Türkiye' de genişlemeye başlamıştır.Bu şirketler genel olarak duyulan gereksinimi karşılamla beraber o dönemde sigorta sektörü için kuruluş ve faaliyetleri düzenleyen devlet denetimi ve kanunların olmayışı nedeniyle denetimsiz bir şekilde ve merkezlerinden aldıkları talimatlara göre çalışıyorlardı. Poliçelerini kendi dillerinde düzenliyorlar, anlaşmazlık durumlarında Londra veya ilgili merkezin bulunduğu yerel mahkemeleri dava mercii olarak gösteriyorlardı. Zamanla açılan Türk şirketleri, yapılan düzenlemeler ve yenilikler ile yabancı şirketlerin Türk şirketleri ile ortaklıklar kurma yoluna gittikleri görülmüştür. Daha sonra 1939 yılında sigorta şirketleri Türk Ticaret Bakanlığına bağlanmıştır. (www.tsb.org.tr, 2013)

Hilenin sözlük anlamı; "Birini kandırmak, yanlış yönlendirmek için yapılan düzenbazlık, oyun, dolap olarak tanımlanır". Bir kişiden herhangi bir konuda beyanda bulunmasını ve ilgili kişiyi bu konuda anlaşma/sözleşme yapmaya yönlendirmek için, bilinçli olarak yanlış ve hatalı bir eylemi gerçekleştirmesine ve sürdürmesine "hile" denir.

Sigorta hileleri son yıllarda dikkat çeken bir konu haline gelmiş olmakla birlikte üzerinde çok fazla sayıda araştırma yapılmaya başlanmıştır. Sigorta hileleri ise karşımıza daha çok poliçe sahibinin, uğradığı zararın tutarı ile ilgili yanlış bildirimde bulunması, gerçekleşmemiş, olmamış bir hasarla ilgili tutanak tutmak ve bildirimde bulunmak, ayrıca bir sigorta hizmetinin başlangıcında sigortalı tarafından bildirilmesi gereken bilgilerin beyan edilmemesi veya yanlış bilgi verme gibi meydana gelmiş zararın miktarını kasıtlı olarak büyütme gibi yollarla yapılan hilelerdir. (Derrig, 2002) Sigortalı kişileri sigorta hilelerini yapmaya yönlendiren temel motivasyon kaynağı mali olarak sağlayacağı fayda olduğu açıkça görülmektedir. Sırf bu faydayı sağlayabilmek için kişiler bilinçli ve kasıtlı bir şekilde veya bazı durumlarda istemsiz olarak masum olan kişilere zarar vermektedirler. Kasıtlı yapılan zararlara örnek, hayat sigortası olan bir kişinin bu sigorta tazminatından fayda sağlamak için sigortalı şahsın kasıtlı olarak öldürülmesi verilebilir. Sigortaya konu olan her şey aynı zamanda sigorta hilesine konu olabilir. Örnek; sağlık sigortaları, hayat sigortaları, işçi tazminat sigortaları, konut ve eşya sigortaları, araç sigortaları sigorta hilesine maruz kalabilmektedir. (Mengi, 2014)

Sağlık sektörü için hile, kasıtlı bir şekilde kandırma ve gerçeğin farklı bir şekilde gösterilerek haksız bir kazanç sağlama anlamına gelmektedir. Gerçekte sigortalının kullanmadığı sağlık hizmetlerinin sanki kullanılmış gibi gösterilerek fayda sağlanması bu durum için iyi bir örnektir. Sahtecilik ve hile haricinde sağlık sektörü için ilaveten sigortanın kötüye kullanımı da sıkça görülmektedir. Örnek olarak sigortalının ihtiyacı olmadığı halde bazı sağlık hizmetlerini kullanması veya kişiye ihtiyacı olmadığı halde gereksiz laboratuvar işlemlerinin yapılması gösterilebilir (www.quackwatch.com, 2012)

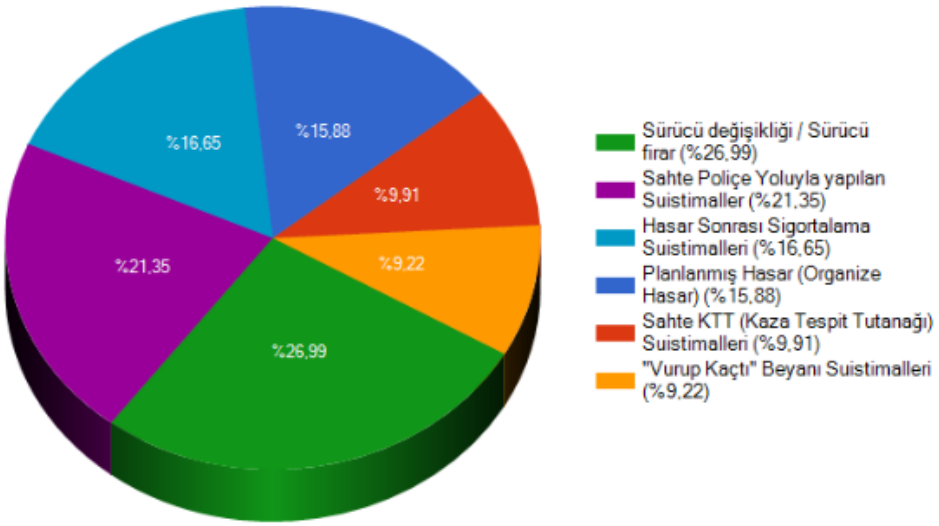
Hayat sigortalarında bilindiği üzere sigortalılardan alınan belirli miktarlarda ödemeler ile bir fon oluşturulmaktadır. Örneğin veriler 30 yaşına girmiş olan erkeklerden bir yıl içinde bir kişinin ölme ihtimali bulunduğunu belirtiyor ise, sigortalı herkesin 1TL vermesi ile toplamda oluşan 1.000 TL, ölen sigortalının poliçesindeki lehtara ödenmektedir. (Brown, 1964) Bu durumda, bir sigorta poliçesinin lehtarı olmak, sigortalanan kişinin ölmesi durumunda lehtarın bir tazminat alacağının göstergesidir. Sigortalıların aile fertlerini koruması için çok iyi bir teminat olan hayat sigortası, sahtekarlık yapma niyeti olan insanların elinde çok kötü amaçlarla kullanılabilir. Hayat sigortalarında yapılan sahteciliklere örnek olarak kişilerin aile bireylerinin sigorta

şirketinden yüklü miktarlarda para alabilmesi adına ölü numarası yapmaları verilebilir. Hatta bazı kişilerin sonradan öldürmüş gibi gösterebilecekleri ve sigortadan yüklü tazminat alabileceği sahte kimlikler yarattığı da görülmektedir. Bu iki tür hilede de kişiler fiziksel olarak zarar görmemektedir. (www.wisegeek.com, 27.06.2012) Sigorta hilelerine verilecek örnekler ilgili sigorta branşına göre çoğaltılabilir. Örnek olarak; işçi tazminat sigortası hilelerinde çok karşılaşılan durum çalışanların iş dışında olan yaralanma olayı ile karşılaşmaları durumunda ortaya çıkmaktadır veya konut ve eşya sigortalarında ilgili varlıklara kasıtlı olarak zarar vermek suretiyle tazminat almak. (www.insurancefraud.org, 2013) (Loughran, 2005)

Sigorta hilelerinden doğan gerçek zarara ilişkin net bir tutarın hesaplanabilmesi bazı durumlarda özellikle soygun, cinayet gibi net maliyet hesabı yapılamayan nedenlerden ötürü çok zordur. Pek çok sayıda sigorta hilesi gerçekleşmiş olmasına rağmen fark edilmeden veya etkisi hesaplanamadan kalır. Sigorta Hileleri Karşıtı Koalisyon isimli kuruluş (Coalition Against Insurance Fraud), ABD'de 2006 da gerçekleştirdiği araştırmada sigorta sahtekarlıklarının sektöre verdiği zararların yaklaşık 80 milyar dolar civarında olduğunu göstermiştir. (Coalition Against Insurance Fraud Annual Report, 2006)

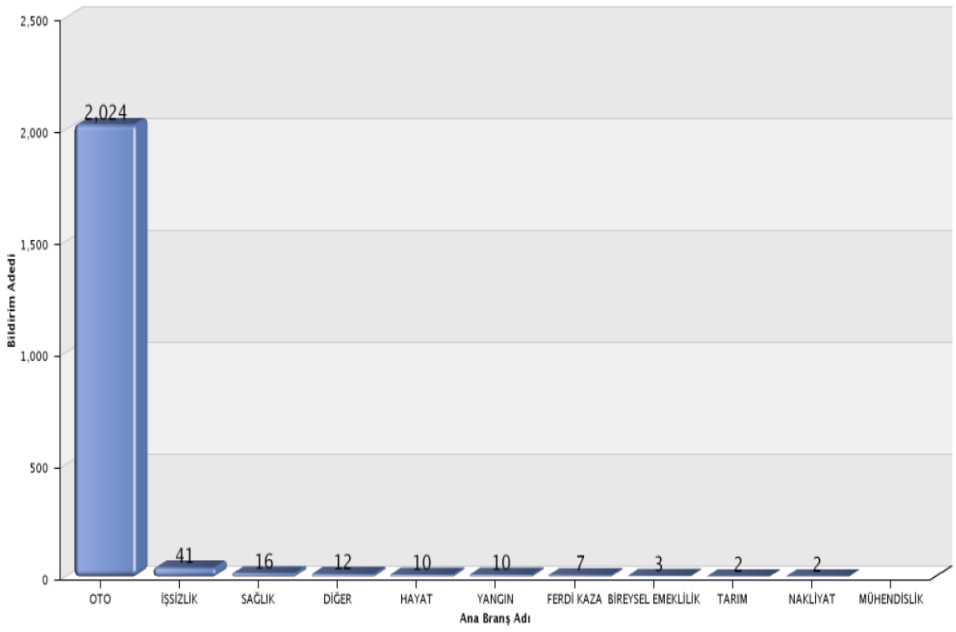
1.4 Sigorta Sektöründe Sahteciliğe Bakış ve Rakamsal Veriler

Dünyada olduğu gibi Türkiyede de sigorta suistimalleri çokça ve değişik yöntemlerle yaşanmaktadır. Sigorta suistimallerinde gözlenen artış ve dürüst sigortalıları korumak için Hazine Müsteşarlığına bağlı çalışan Sigorta Bilgi Merkezi Aralık 2009 da Sigorta Suistimalleri Bilgi Sistemi (SİSBİS) projesini hayata geçirmiştir. Bu sistemin gayesi sigorta kurumları için sahte hasar ilave maliyetlerinin düşürülmesi, risklerin kapsam altına alınıp alınmayacağı ve risklerin fiyatlarının belirlenmesi konularında daha doğru kararların alınması sayılabilir. Sigortalılar için ise dürüst sigortalıların haklarının korunması ve buna bağlı olarak sigorta prim maliyetlerinin düşürülmesidir. (siseb.sbm.org.tr, 2020) SİSBİS istatistiklerine göre Haziran 2019 ve Haziran 2020 arasında bildirlen suistimal yöntemleri şekil 1 de gösterilmiştir. (https://siseb.sbm.org.tr, 2020)



Şekil 1: Suistimal Yöntemleri (Haziran 2019 – Haziran 2020)

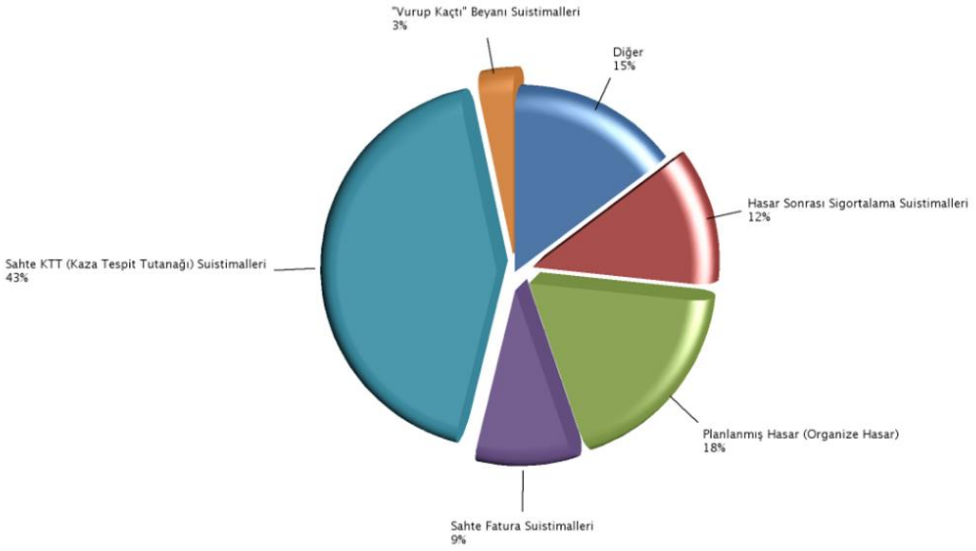
SBM den alınan verilere göre branş bazında suistimal dağılımları şekil 2 de gösterilmiştir. Bariz bir şekilde oto branşında çok daha fazla suistimal bildirimi yapıldığı görülmektedir.



Şekil 2: Branş Bazında Suistimal Dağılımı(*)

(*)-Bildirilen toplam adet: 2.127'dir -01/01/2014 30/09/2014 Tarihleri Arasındaki Verilerin Kümülatif Toplamını Yansıtmaktadır.

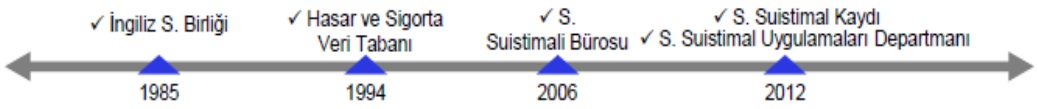
SBM verilerine suistimal türlerine ait bilgi şekil 3 te verilmiştir. En çok sahte kaza tespit tutanağı düzenlenmesi yöntemi ile sahteciliğe başvurulduğu görülmektedir.



Şekil 3: Suistimal Türleri

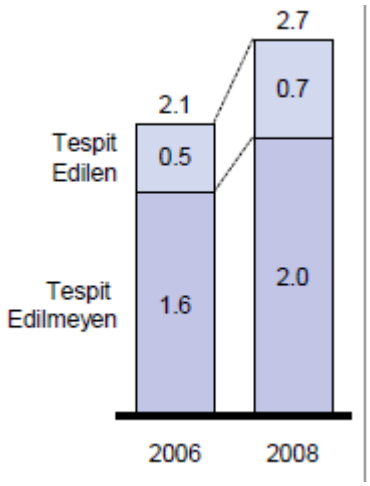
1.5 Dünya' da Sektöre Genel Bakış Ve Rakamsal Veriler

İngiltere'de uygulanan sistemin temelini Suistimal Bürosunun sağladığı kurumlar arası koordinasyon ve ortak veri paylaşımı oluşturmaktadır. Altta İngiliz sigorta v esuistimal alanında gelişmeleri şekil olarak bulabilirsiniz.



Şekil 4: İngiliz Sigorta Suistimal Gelişimi

İngiliz Sigortacılar Birliği (ABI) 400 üyesi ile İngiltere piyasasının %94 sigorta hacmini, 200 milyon pound yatırımı ve 32 milyonunun üzerinde hasar dosyası ile geçmiş verileri ve işlemleri yönetmektedir. IFB sigorta suistimalleri bürosu gelen ihbarları değerlendirip, suistimal durumlarında sigorta şirketlerine ve gerekli mercilere bildirimleri yapmaktadır.



Şekil 5: İngiltere Suistimal Miktarları (Milyar £)

Kaynak : ABI

Amerika'da suistimalle mücadelede eyalet destekli kuruluşlar ve sigorta şirketleri arasındaki düzenli örgütlenme öne çıkmaktadır. Sigorta şirketleri yapıları içerisinde bulunan suistimal sistemleri ile doğrudan müşteri iletişim sağlayabiliyorlar. Yine Amerika' da Ulusal Sigorta Suçları Bürosu (NICB), Sigorta Suistimalleri ile Mücadele Komisyonu (CAIF) ve Ulusal Sigorta Komisyonerleri Birliği (NAIC) bu alanda faaliyet gösteren en önemli kurumlar olarak sayılabilir.

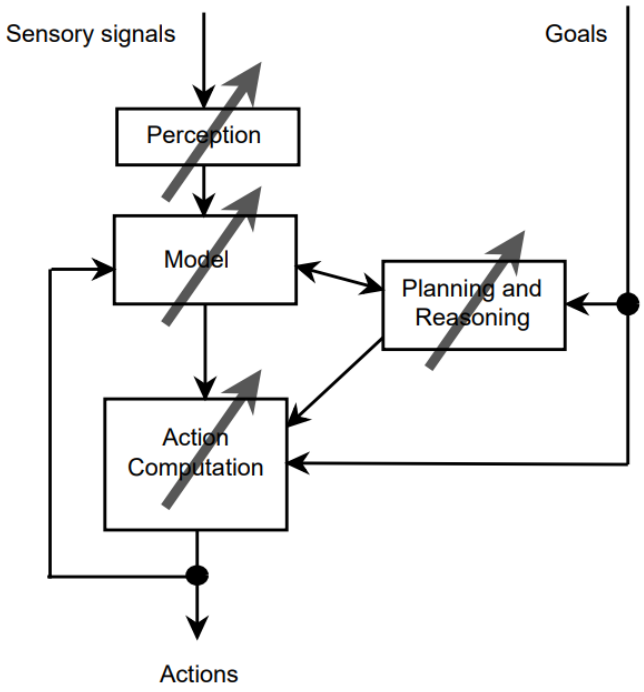
Küresel anlamde sigorta sahtekarlığını tespti etmek için oluşan market yaklaşık olarak 3,29 milyar ABD doları tutarunda olduğu düşünülmekte ve 2027 ye kadar %15 büyümesi öngörülmektedir. (www.businesswire.com, 2019) Bu da bize sahtecilkik tespiti için sigorta sektörünün yaptığı ciddi yatırımları ve sahteciliğin sigorta sektörüne global anlamda ne kadar zarar verdiğini, önlenmesi için ciddi yatırımların yapıldığını göstermektedir. Tüm bunlar göz önüne alındığında sigorta şirketlerinin sahteciliği tespit etmede geleneksel yöntemlerden kurtulup daha modern ve gelişmiş önleme, tepit etme tekniklerini benimsemeye başlaması gerekmektedir. Veri analitiğine yapılan yatırımlar sigorta sahtekarlığı tespit pazarını da etkilemiş ve daha iyi şekillendirerek dünyadaki talebini de arttırmıştır. Bu segment ayrıca sahtekarlık analizi, kimlik doğrulama, yönetişim, risk ve uyumluluk gibi sigorta sahtekarlığı tespit endüstrisindeki çözüm ve hizmet sağlayıcıların müşteri deneyimini arttırmaya yardımcı olduğu gibi BYOD ve IoT gibi dijital teknolojilerin artan kullanımına istinaden karşılaşılan güvenlik tehditleri ve zorluklarıyla başa çıkmalarına da yardımcı olur. Buna ek olarak, coğrafyaya dayalı sigorta sahtekarlığı tespit pazarına 2018

yılında Kuzey Amerika hakim olmuştur. Bölgenin saltanatına devam etmesi bekleniyor. Kuzey Amerika'da ABD, sigorta sahtekarlığı tespit piyasasına hakimdir. Ülkede artan sigorta sahtekarlığı vakaları pazarın büyümesini ve bu alandaki tecrübelerinin artmasına imkan vermektedir. (www.businesswire.com, 2019)

2. MAKİNE ÖĞRENMESİ

2.1 Makine Öğrenmesi Nedir

Makine öğrenmesi, örnek veri setleri üzerinden veya eski deneyimlerin kullanılarak optimum performans ölçütünü elde etmek için bilgisayarın programlanmasıdır. Öğrenme işlemi, eğitime tabi tutulacak verilerin veya eski deneyimlerin kullanılarak oluşturulan modelin parametrelerini optimum seviyelere çıkarmak için bir bilgisayar programının koşturulması olarak tarif edilebilir (Alpaydın, 2020). Makine öğrenmesi ile, geçmiş tecrübelere dayanarak veya örnek veri setlerine dayanan yeni işlemleri optimize etmek için bilgisayarları programlayabiliriz. Buna göre ilgili sınıflandırma işlemleri bilgisayarda çok kısa sürelerde ve etkili bir biçimde yapılabilir, sonucunda uygun bir model oluşturulur ve bu model bize geleceğe yönelik öngörülerde bulunabilir, denetim amacıyla da kullanılabilir. (Kızılkaya ve Oğuzlar, 2018) Öğrenme sözlük anlamına bakıldığında kavramsal düzenlemeler yapma, alıştırma ve uygulamaların etkileri ve bilgi, beceri, anlayışlar edinme gibi ifade edildiği görülmektedir. Zoolog ve psikologlar insanlar ve hayvanlar üzerindeki öğrenme ile ilgili çalışmalar yapmaktadırlar. Buradan makine öğrenimi ile canlıların öğrenmesi arasında çok fazla paralellik olduğunu görebiliriz. Zaten bir çok makine öğrenmesi tekniği psikologların hayvan ve insan öğrenme teorilerini araştırma çabalarından kaynaklanmış, esinlenmiştir. Makine öğrenimi genellikle yapay zeka ile ilişkili görevleri yerine getiren sistemlerde meydana gelen değişiklikleri ifade eder. Bu tür görevler, tanıma, tanılama, planlama, robot kontrolü, tahmin vb. Biraz daha spesifik olmak gerekirse, Şekil 6' daki tipik bir AI "ajan" mimarisini gösteriyoruz. Bu ajan çevresini algılar ve modeller ve belki de etkilerini tahmin ederek uygun eylemleri hesaplar. Şekilde gösterilen bileşenlerden herhangi birinde yapılan değişiklikler öğrenme olarak sayılabilir. Hangi alt sistemin değiştirildiğine bağlı olarak farklı öğrenme mekanizmaları kullanılabilir. (Nilsson, 1996)



Şekil 6: AI Sistem Ajan Mimarisi

2.2 Makine Öğrenmesi Yöntemleri

Makine öğrenmesinin büyük veri tabanlarına uygulanmasına veri madenciliği denilmektedir. Analoji, az miktarlarda olan fakat işlendiğinde çok değerli malzemeye ulaşmamıza imkan sağlayan büyük miktarlarda toprak ve hammaddenin bir madenden çıkarılmasıdır; veri madenciliğinde, örneğin yüksek tahmin doğruluğu olan değerli bir kullanımla basit bir model oluşturmak için büyük miktarda veriye ihtiyaç duyulur ve işlenir. Çok fazla uygulama alanı vardır; örnek olarak perakendeciliğe ek, finans alanında bankalar geçmiş verilerini ve kredi uygulamalarında kullanmakta, ayrıca sahtekarlık tespiti ve borsada kullanımı için yeni modelleri geliştirmek amacıyla analizler yapıldığı görülmektedir. İmalat sektöründe, öğrenme modellerinin optimizasyonu, kontrol ve sorun giderme için kullanıldığı görülmektedir. Tıpta, öğrenme programları tıbbi teşhis için kullanılmaktadır. Telekomünikasyonda, ağ optimizasyonu ve hizmet kalitesini en üst düzeye çıkarmak için çağrı modelleri analiz edildiğini görmekteyiz. Bilimde fizik, astronomi ve biyolojideki büyük miktarda veri ancak bilgisayarlar tarafından yeterince hızlı analiz edilebilmektedir. World wide web bilindiği üzere çok büyük ve sürekli büyüyor. İlgili bilgileri aramak maalesef elle ve manual yapılacak bir işlem değildir. Makine öğrenmesi veri madenciliğinde olduğu gibi sadece veritabanı problemleri ile ilgilenmez. Ayrıca bir yapay zekaya sahiptir.

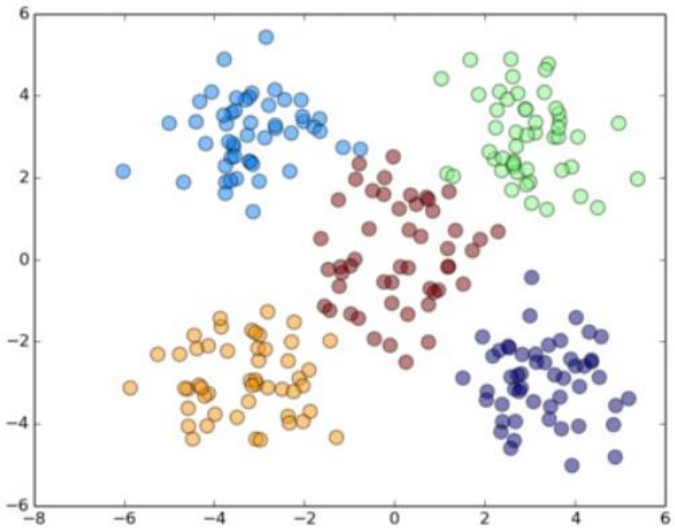
Zeki olabilmesi içinde değişen bir ortamda sistemin öğrenme yeteneğinin olması gerekir. Eğer sistem düzgün bir şekilde öğrenebiliyorsa dizayn eden kişinin bulunmasına gerek almadan bütün olasılıklar için çözümler sunabilir (Alpaydın, 2020).

2.2.1 Denetimli öğrenme

Denetimli öğrenme modeli birbiriyle etkileşimi olan veri setleri üzerinden belli düzenlerde organize edilerek gerçekleştirilen öğrenme yöntemidir. Başka bir şekilde tarif etmek gerekirse makine öğrenimini verilen girdilere göre bir dizi çıktılar alınan karakutu olarak düşünebiliriz. Verilen girdilere göre alınan çıktılar kümesi için varsa elimizdeki geçmiş verilere dayanan öğrenme modeline denetimli öğrenim denmektedir. Çok bilinen denetimli öğrenim modelleri sınıflandırma modelleridir. (Joshi, 2019).

2.2.2 Denetimsiz öğrenme

Veriler arasında belirgin bir ilişki bulunmayan ve sınıf bilgisi içermeyen verilerin içerdiği grupların incelendiği yöntemdir. Denetimsiz öğrenme modellerinde etiketlenmiş veriler bulunmaz. En klasik ve bilinen örnek kümeleme algoritmalarıdır (Joshi, 2020). Denetimsiz öğrenme veri kümeleri içinde verileri yorumlama yaparak ortak yönlerini bulmaya ve bu verileri kümeleme yaparak anlamlı bir veri elde etmeye çalışır. Verilerin belirli özelliklerine göre benzerlik, yakınlık, uzaklık vb yönlerine göre analiz ederek sınıflanmasına kümeleme denir.



Şekil 7: Kümeleme

2.2.3 Yarı denetimli öğrenme

Yarı denetimli öğrenme tam olarak denetimli ve denetimsiz öğrenme methodlarının arasında yer alır ve bibiriyle etkileşimi bulunan büyük veri ile belirgin bir ilişkisi bulunmayan düşük miktarda verinin birlikte kullanıldığı bir yöntem olarak tanımlanabilir. Yarı denetimli öğrenme için diğer algoritmalarda olduğu gibi bilgisayarların ve doğal sistemlerin etiketlenmiş ve etiketlenmemiş veriler ışığında nasıl öğrendiklerine ilişkin bir öğrenme paradigmasıdır. Klasik olarak öğrenmenin tüm verilerin etiketsiz olduğu denetimsiz öğrenme (örnek: aykırı tespit, kümeleme) veya tüm verilerin etiketli olduğu denetimli öğrenme (örnek: regresyon, sınıflandırma) olarak incelendiği görülmektedir. Yarı denetimli öğrenme modelinin amacı ise etiketli ve etiketsiz olarak birleştirilmiş verilerin öğrenme davranışlarını ne şekilde değiştireceğini anlamak ve bu kombinasyonlardan faydalanan algoritmaları tasarlamaktır. (Zhu ve Goldberg, 2009).

2.2.4 Takviyeli öğrenme

Takviyeli öğrenme çıktı olarak doğru veya yanlış şeklinde sonuçlar alınan bir yöntemdir. Ayrıca pekiştirmeli öğrenme olarak isimlendirilen bu yöntem sebep sonuç ilişkisine dayalı bir öğrenmedir ve denetimli denetimsiz öğrenme yöntemlerine göre farklı olarak incelenmesi gerekir. Takviye veya pekiştirme öğrenimleri çevreden alınan geribildirimlerden oluşur ve tam olarak denetlenemez. Özetle bu yöntemde eğitim için kullanılan veri etiketli veri dizisi değildir (Joshi, 2020).

2.3 Makine Öğrenmesi Algoritmaları

2.3.1 Random forest (rasgele orman)

Rasgele Orman algoritmaları sınıflandırma işlemleri yapılırken değerleri yükseltmek için birden çok karar ağacı oluşturur ve torba karar ağaçlarına göre daha geliştirilmiş versiyonudur diyebiliriz. CART (Classification and Regression Trees) karar ağacları algoritmalarının sorunu aç gözlü bir algoritma olmalarıdır. Aç gözlü algoritmalar da fazlaca korelasyona sahiptirler ve bu algoritmanın zayıf yönüdür. Bu nedenle alt

modellerden üretilen çıktılar zayıf korelasyona sahip olabilir. Rasgele orman algoritması bu şekilde alt ağaçlardan gelen tahminlerin zayıf olmaması ve az korelasyonda olması için ağaçların öğrenme şekline ilişkin algoritmayı değiştirmeye çalışır (Oshiro,2012). Rastgele orman algoritması, sınıflama ve regresyon işleri için eğitim aşamasında fazlaca karar ağacı oluşturur ve problem tipine göre sınıflandırmayı veya regresyonu yapan toplu öğrenme algoritmasıdır.

2.3.2 Linear regression (doğrusal regresyon)

Linear Regression algoritması tahmin edeceği değişken için başka değişkenlerin değerlerini kullanan bir yöntemdir. Tahmin edilen değer bağımlı değişken olarak adlandırılırken, tahmin için kullanılacak diğer değişkenler bağımsız değişken olarak isimlendirilir. Bu yöntemde, tahmin edilmek istenen değişken için değeri en iyi tahmin eden bağımsız değişkenler kullanılır ve doğrusal denklem için katsayıları belirler. Doğrusal regresyon, tahmin edilen ve doğru değerler arasında uyumsuz durumları minimuma indiren doğrusal bir çizgi veya yüzeye konumlanır. Bu algoritmada eşleştirilen bir çift veri kümesi içinde en uygun değeri içeren satırı bulmak üzere "en küçük kareler" yönteminin kullanıldığı basit doğrusal regresyon hesaplama araçları kullanılmaktadır. Bu araçlar vasıtasıyla y bağımsız değişkeni kullanılarak x bağımlı değişkeninin değerini tahmin edebiliriz (www.ibm.com, 2020). Özetle doğrusal regresyon methodu x giriş değişkeni ve tahmin edilen çıkış y değişkeni arasındaki ilişkiyi altta verilen denklemde olduğu gibi doğrusal bir şekilde tanımlar (Joshi, 2020).

$$y = 0\_1 + 0\_2.X$$

2.3.3 Logistic regression (lojistik regresyon)

Logistic Regression algoritması ikili değere sahip verilerin çıktılarını hesaplamak için lojistik fonksiyonların kullanıldığı makine öğrenimi methodudur. İkili değerlere sahip verilere örnek doğru veya yanlış, 0 veya 1, evet ya da hayır gibi çıktılar veren değişkenler örnek olarak gösterilebilir. Lojistik regresyon algortiması y değişkeni değerinin olasılığı ile verimizdeki x değişkeni öznitelikleri arasında bulunan ilişkiyi belirlemeye yardımcı olmaktadır. Lienear regression algoritmasını incelediğimizde -∞, +∞ aralığında sonuçlar alındığı görülür. Bu sonuçların olasılık değeri elde etmeye çalıştığımız için 0 – 1 aralığında bir değere çevrilmesi gerekir. Bu işlem için sigmoid fonsiyonu kullanılır (Joshi, 2020).

$$P(y = 1|x)$$

$$\sigma(x) = 1/(1 + e\text{^}(-x))$$

2.3.4 K nearest neighbors (k-en yakın komşular)

En yakın komşular algoritması mevcut veriler üzerinden yeni vakaları benzerliğine göre sınflandırır. Bu algoritmada vaka, komşuların oylarının çokluğu ile sınıflandırılır ve mesafe fonksiyonu kullanılarak yapılan ölçümle en yakın komşular içinde yaygın bir sınıfa atanır. K = 1 olduğu durumda olay en yakındaki komşunun sınıfına atanır. K En yakın komşular algoritması sınıflama hesaplamalarını Manhattan, Öklid ve Minkowski mesafe fonksiyonlarını kullanarak benzerlik ölçüsüne dayalı bir şekilde gerçekleştirir (Hazım, 2018).

2.3.5 Naive bayes

Naive Bayes modeli adını İngiliz matematik bilimcisi Thomas Bayes'ten alır. Naive Bayes algortiması örüntü problemlerinde kullanılan kısıtlayıcı gibi görünen olasılıkcı bir önerme methodudur. Bu önermede, örüntüleri tanımlamada kullanılan tanımlayıcılar, öznitelikler veya parametrelerin istatistiki yönden bağımsız olmaları gereklidir. Bu önerme Bayes sınıflandırmaların kullanım alanını kısıtlamasına rağmen istatistiki bağımsızlık koşulları esnetilip kullanılarak karmaşık yapay sinir ağları ile karşılaştırılabilecek seviyede sonuçlar verdiği görülmektedir (Aggarwal, 2014:67). Bayesian olasılık formülü altta gösterilmiştir. Bu denklemde P(Y|X) hipotezin olasılık yani sahte hasar olup olmadığını gösterir. P(Y) ise hipotezin eski deneyimlerden sahip olduğu olasılıkları göstermektedir. Son olarak denklemdeki P(X|Y) ise olasılığın oranını gösterir (Hazım, 2018).

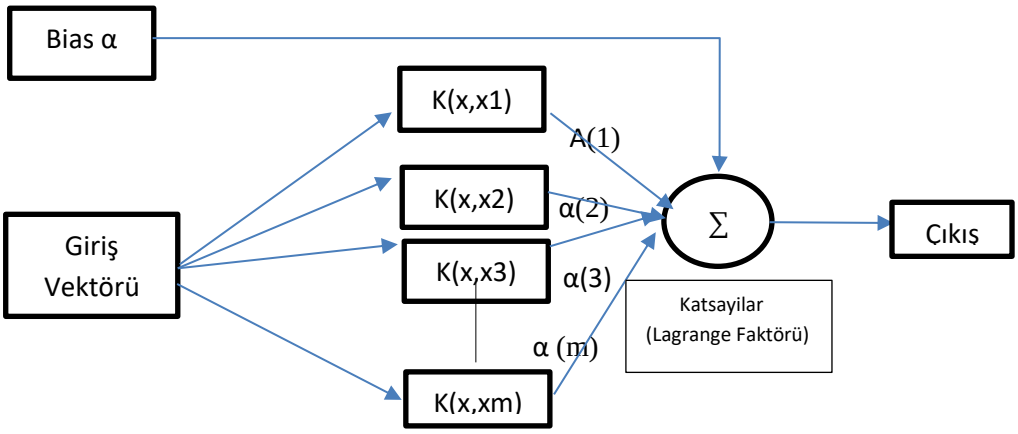
$$P(\mathrm{Y}|\mathrm{X}) = P(X|Y).\,\mathrm{P(Y)}|\mathrm{P(X)}$$

2.3.6 Decision trees (karar ağaçları)

Karar ağaçları algoritmalarında veriler belirlenmiş bir parametreyle düzenli ve sürekli bölünerek değerlerin elde edildiği denetimli bir makine öğrenmesi algoritmasıdır. Karar ağaçları algoritması üç yapısal bölümden oluşur. Nodes, edges/branch ve leaf nodes. Karar ağaçları genel olarak ikiye ayrılmaktadır, classification ve regression. Bu iki ana sınıftan classificaiton ağaçları ikili değere sahip yes/no gibi sonuçlar verir, regression ağaçları da değişkenlerin sürekli olarak değerler aldığı bir sınıflandırma yöntemidir (Barros, 2015).

2.3.7 Support vector machines (destek vektör makineleri)

Destek Vektör Makinesi algoritmaları kontrollü sınıflandırma yöntemini kullanan istatistiki öğrenmeye dayalı bir algoritmadır. SVM algoritması matematiksel denklemleri ikili sınıfları ve doğrusal verilerin sınıflandırılması problemleri için başlangıçt tasarlanmış olsada sonrasında çok sınıflı, doğrusal olmayan verilerin sınıflandırılması için kullanılımı genelleştirilmiştir. SVM' nin sınıflandırma işlemleri ve örüntülerin tanımlanması problemlerinde Vapnik' in geliştirdiği ve Vapnik Chervonenkis teorisi olarak isimlendirilen teoreme dayanır. Şekil 7' de gösterdiğimiz ağ denklem yapısında K(x,x1) değişkenleri bu fonsiyonun çekirdeğini ve α ise lagrange çarpanlarını göstermektedir. Çekirdek fonksiyon bize girdilerin iç çarpımlarını bulmada yardımcı olur ve α Lagrange çarpanları da denklemdeki ağırlıkları ifade eder. Bu ifadelerden faydalanarak SVM ile elde edilen örnek bir çıktı değeri için, girdilerin iç çarpımlarının ve lagrange çarpanları kombinasyonları toplamı ile elde edilen sonuca eşit diyebiliriz (Ayhan ve Erdoğmuş, 2014).



Şekil 8: Destek Vektör Makinesi Yapısı (Ayhan & Erdoğmuş, 2014).

3. PROJE

3.1 Amaç

Tezimin amacı, oto hasar sigortalarının bildirimlerinde gerçek dışı yani sahte olan hasarları makine öğrenmesi algoritmalarından faydalanarak doğru bir şekilde tahmin etmek ve kullanılan algoritmaları doğruluk oranlarına göre karşılaştırmaktır. Geliştirdiğim uygulamada makine öğrenmesi algoritmalarını elde edilen doğru tahmin oranları ve işlem sürelerine göre karşılaştırdım ve en iyi sonuçların elde edildiği algoritmaları tablo ve grafikler yardımıyla gösterdim. Uygulamamda veri seti içinde önceden belirlenmiş bulunan sahte hasarlar üzerinden öğretim gerçekleştirilip kalan veri üzerinden de test işlemleri yapılarak algoritmaların sırayla sahte hasarları ne oranda doğru tespit ettikleri ölçülmüştür. Ayrıca her algoritmanın işlem süreleri hesaplanarak algoritmalar işlem performanslarına göre karşılaştırılmıştır. Uygulamada özel bir sigorta şirketinden araç sigortaları hasar kayıtları ile hazırlanmış gerçek hasar verilerinden oluşan bir veri seti kullanılmıştır.

3.2 Veri Seti

Tez çalışmamda kullanılan veri seti özel bir sigorta şirketinin hasar birimi tarfından hazırlanan ve oto hasarları için sahte bildirimlerin tespitine özel bir çalışmada kullanılan gerçek bir veri setidir. Sigorta şirketine bildirilmiş veri havuzunda bulunan ve rasgele seçilmiş 3000 adet kayıttan oluşan veri setinde bazı değişkenler ilgili birim tarafından belirlenmiş kural setlerini ifade etmektedir. Normal bir hasar gerçekleştiğinde bu hasara ilişkin sigortalı ve hasarın detayları ile ilgili çok fazla bilgi alınmaktadır. Dolayısıyla veri setimiz normalize edilmiş ve uygulamanın anlayacağı bir hale dönüştürülmüştür. Veri setimiz hasar bildirimlerine ilişkin hasarın sahte olmasını belirlemede kullanılacak kritik verilere ve hasarın sahte mi yoksa gerçek hasar mı olduğunu belirten bilgileri barındırmaktadır. Veri setimizde bulunan bilgilerin değişken isimleri; fraud, dosya_no, teminat_bedel, hasar_neden_aciklama, ozel_tuzel, yeni_is_yenileme, model_yasi, kullanim_tarzi, sig_cinsiyet, sigortali_il,

Öznitelik_1, Öznitelik_3, Öznitelik_6, Öznitelik_12, Öznitelik_13 ve sig_yas. Öznitelik_X şeklinde isimlendirilen kural setlerinin açıklamalarını Tablo 1 de bulabilirsiniz.

Tablo 1: Veri Seti Kural Açıklamaları

Kural	Açıklama
Öznitelik_1	Kanda bulunan alkol miktarı. 0 alkol alınmadığını gösterir.
Öznitelik_3	Geçmiş hasar bilgisi. 1 eski incelenmiş hasar var demektir.
Öznitelik_6	Sahip olunan sigortalı araç sayısını gösterir.
Öznitelik_12	Poliçedeki hasar sayısı.
Öznitelik_13	Hasarın gerçekleşme tarihi ile poliçenin yapıldığı tarih arasındaki fark.

3.3 Yöntem

Tez çalışmamda kullandığım uygulamamı python dilinde geliştirdim. Pyhton kütüphanelerini kullanarak makine öğrenmesi algoritmaları çalıştırılmış ve sonuçları karşılaştırılmıştır. Veri seti kısmında belirttiğim ve detaylarını paylaştığım üzere sektörde faaliyet gösteren özel sigorta şirketinden temin edilen gerçek hasar bilgileri kullanılmıştır. Uygulamayı geliştirirken editör olarak Visual Studio Code kullandım. Ayrıca debugging işlemlerini kolay gerçekleştirebildiğim için ve uygulamayı adım adım koşturup çıktılarını görebildiğim için Jupyter Lab editöründen de faydalandım. Uygulamada doğruluk oranları ve test süreleri performansları karşılaştırılmış olan algoritmalar; rasgele orman, lojistik regresyon, doğrusal regresyon, k en yakın komşular, naive bayes, karar ağaçları ve destek vektör makineleri algoritmalarıdır. Algoritmaların uygulamada kullanımları varsayılan parametreleri ile yapılmıştır. Üç bin adet kayıtlı hasar verisiyle oluşturulmuş veri setimizin eğitim ve test sınıflandırması için iki yöntem kullanılmıştır. Train-test split yöntemi ve k parçalı çapraz doğrulama yöntemi kullanılmıştır (K-Fold Cross Validation). Standart bir yöntem olarka train-test split yöntemi kullanılarak veri test ve train için bölünmüş ve bunlar üzerinden uygulama çalıştırılarak çıktılar alınmıştır. İkinci yöntem ise daha sağlıklı bir test imkanı sunan k-fold cv yöntemidir ve bu yöntemde veri setimiz önceden belirlediğimiz sayıda eşit ve değişik kümelere ayrılır ve uygulamada bu kümelerin hepsi teste tabi tutulmuştur. Bu yöntem veri setimizi k adet eşit parçaya böler ve her seferinde farklı bir küme test için ayrılır ve kalan k-1 adet veri kümemiz

uygulama tarafından eğitim amacıyla kullanılır. Bu işlem k defa tekrarlanır ve her seferde test edilen küme değiştirilir ve sınıflandırıcı eğitilmiş olur (Narin, 2014). K parçalı çapraz doğrulama dizinleri için ayrı ayrı sonuçları "k-CV Değerleri" olarak görünecek şekilde her algoritma için aşağıda paylaşılmıştır. Uygulama ayrıca algoritmalar için işlem sürelerini de doğruluk oranları ile birlikte hesaplayacak şekilde geliştirilmiştir. İşlem sürelerinin ölçümü için her algoritmanın işleme başlaması esnasında ve işlemin bitiş anında atanan iki zaman değişkeni arasındaki fark çıkarılarak yapılmıştır. Uygulama doğruluk oranlarını tahmin edilen hasar verilerinden fraud olanları doğru tespit etmesi üzerinden ölçülmüştür. Kullanılan bütün algoritmalara fraud olan hasar kayıtları için bir tahmin yaptırılmış ve elde edilen sonuçlar karşılaştırılmıştır.

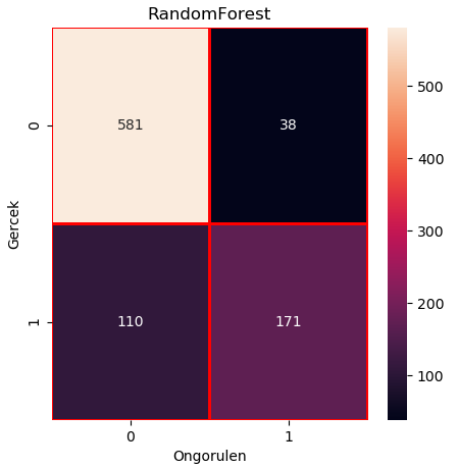
3.4 Sonuçlar

Uygulamamızı test-train split metodu ile çalıştırdığımızda algoritmalara göre elde edilen sonuçlar aşağıda verilmiştir. Bu yöntemde 3000 kayıtlık veri setinin %30 kısmı test için kullanılmış olup kalan veri öğrenme için kullanılmıştır. Rasgele Orman algoritması ile yapılan çalışmada varsayılan değerler ile alttaki sonuçlar elde edilmiştir.

Başarılı veri adedi : 755

Random Forest doğruluk oranı : 0.84

RF işlem Süresi – sn : 0.041



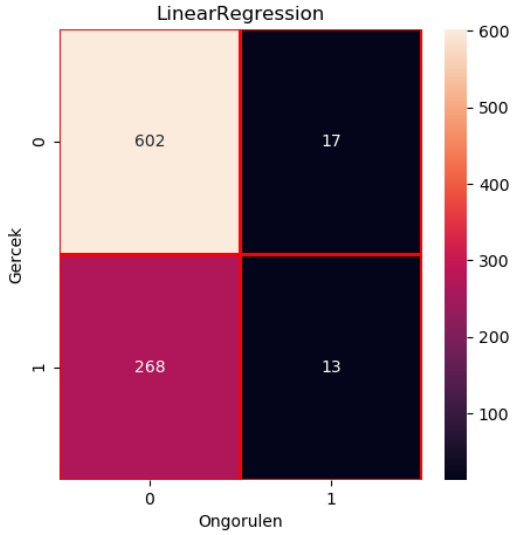
Şekil 9: Random Forest Hata Matrisi (Confusion Matrix)

Linear Regression analizi ile yapılan çalışmanın sonuçları;

Başarılı veri adedi : 615

LR doğruluk oranı : 0.68

LR hesap süresi – sn: 0.006



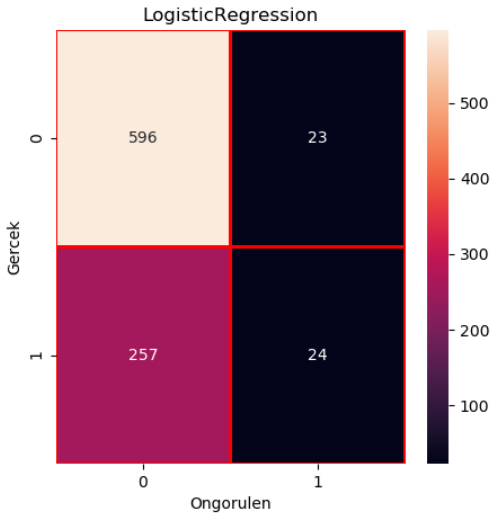
Şekil 10: Linear Regression Hata Matrisi (Confusion Matrix)

Logistic regression analizimizi default parametreler ile çalıştırıldığımız zaman %69 doğruluk oranı skoru alınmıştır.

Başarılı veri adedi : 620

LR doğruluk oranı : 0.69

LR hesap süresi – sn: 0.002



Şekil 11: Lojistik Regresyon Hata Matrisi (Confusion Matrix)

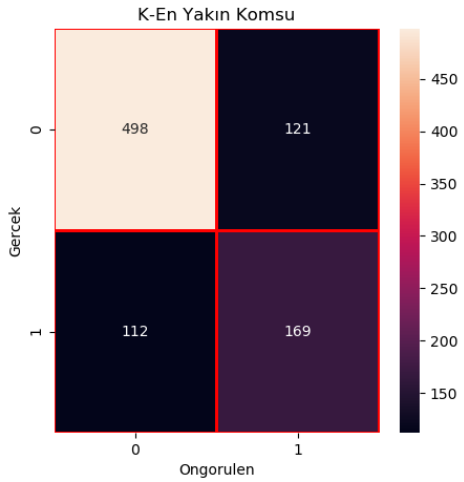
K en yakın komşu algoritmasını çalıştırdığımızda alttaki değerleri elde ettik.

En iyi komşu değeri (k) : 1

Başarılı veri adedi : 667

KNN doğruluk oranı : 0.69

KNN işlem süresi – sn: 0.082



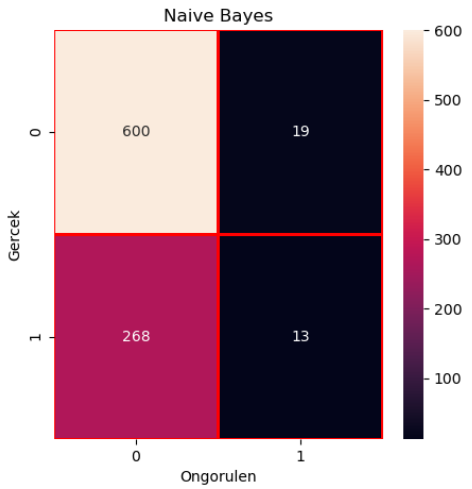
Şekil 12: K-Nearest Neighbor Hata Matrisi (Confusion Matrix)

Naive Bayes algoritması ile yapılan çalışma varsayılan parametreleri ile %68 dogruluk oranı sonucu elde edilmiştir.

Başarılı veri adedi : 613

NB doğruluk oranı : 0.68

NB işlem süresi – sn: 0.003



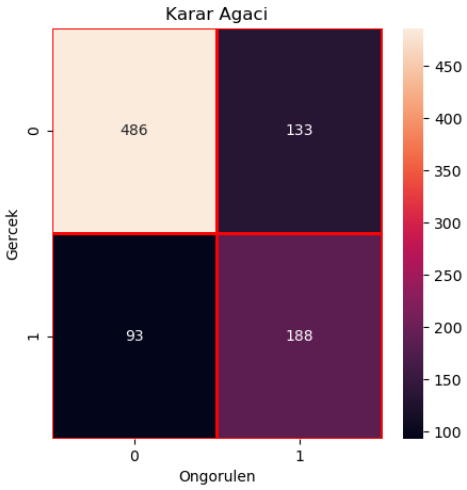
Şekil 13: Naive Bayes Hata Matrisi (Confusion Matrix)

Karar ağacı algoritması çalıştırıldığında alttaki veriler elde edilmiştir.

Başarılı veri adedi : 664

DT doğruluk oranı : 0.74

DT işlem süresi – sn: 0.002



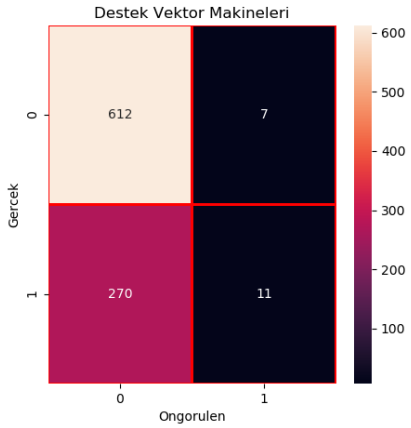
Şekil 14: Decision Tree Hata Matrisi (Confusion Matrix)

Destek Vektör Makineleri algoritması çalıştırıldığında %69 doğruluk oranı görülmüştür.

Başarılı veri adedi : 623

SVM doğruluk oranı : 0.69

SVM hesap süresi – sn: 0.081



Şekil 15: Support Vector Machines Hata Matrisi (Confusion Matrix)

İkinci test ve train modelimiz k parçalı çapraz doğrulama yöntemi ile yapılan test sonuçları ve confusion matrix değerleri de altta paylaşılmıştır. Bu yöntemde bütün veri kümelere ayrılıp test ve train aşamasına sokulduğu için doğruluk oranı veri büyüklüğümüz yani 3000 kayıt üzerinden hesaplanacaktır. Diğer yöntemde olduğu gibi sadece %30 luk kısımdan hesaplanmayacaktır. Sonuçlarda görünen k-CV değerleri her dizin için bulunan doğruluk oranlarını göstermektedir. K değeri olarak 5 kullanılmıştır. Yani veri 5 kümeye bölünerek çapraz doğrulama yöntemi uygulanmıştır.

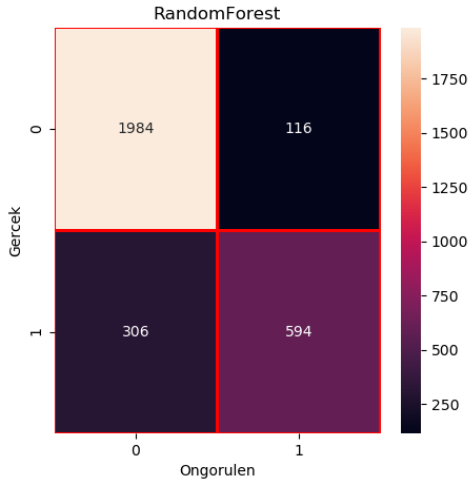
Rasgele Orman algoritması k-CV yöntemi test sonuçları ve confusion matrix;

Başarılı veri adedi : 2578

Random Forest doğruluk oranı : 86%

RF İşlem Süresi – sn : 4.41

RF k-CV Değerleri: [0.87166667, 0.85166667, 0.85666667, 0.845, 0.855]



Şekil 16: Random Forest Hata Matrisi (k-CV Confusion Matrix)

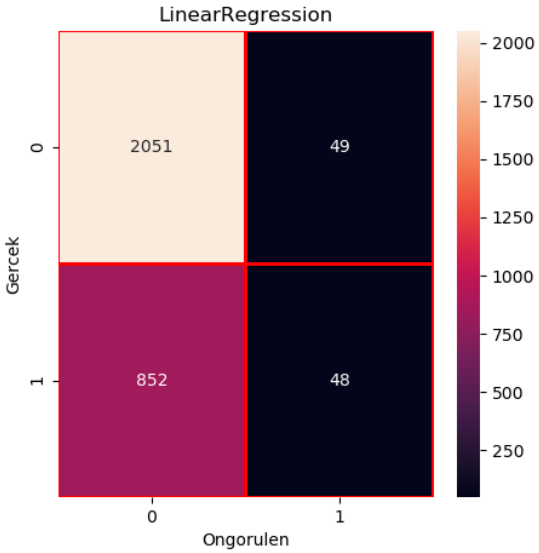
Doğrusal Regresyon algoritması k-CV yöntemi test sonuçları ve confusion matrix;

Başarılı veri adedi : 2099

LN doğruluk oranı : 70%

LN işlem süresi – sn: 0.08

LN k-CV Değerleri: [0.14172893, 0.1927147, 0.19500589, 0.23342316, 0.24026156]



Şekil 17: Linear Regression Hata Matrisi (k-CV Confusion Matrix)

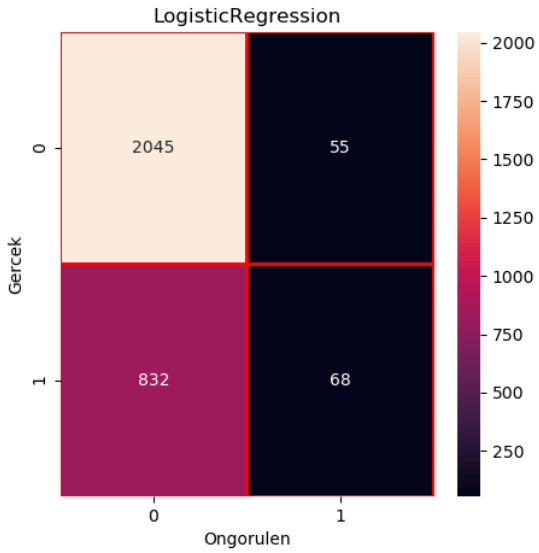
Lojistik Regresyon algoritması k-CV yöntemi test sonuçları ve confusion matrix;

Başarılı veri adedi : 2113

LR doğruluk oranı : 71%

LR işlem süresi – sn: 2.41

LR k-CV Değerleri : [0.72, 0.71333333, 0.70166667, 0.695, 0.69833333]



Şekil 18: Lojistik Regresyon Hata Matrisi (k-CV Confusion Matrix)

K en yakın komşu algoritması ile yapılan çalışmada önemli olan adımlardan biri de en uygun k değerini bulmaktır. Bunun için uygulama üzerinde veri setimizin büyüklüğü kadar tekrar eden bir döngü tanımlanmış ve skorlar dizin şeklinde yazdırılmıştır. Bu dizin içinden en yüksek değeri elde ettiğimiz skor için index değeri alınmış ve k değişkeni için kullanılmıştır. En uygun komşu yani k değeri 1 olarak bulunmuştur. İlaveten alınan k değerini doğrulamak amacıyla k için 1 ile 12 arasında değerleri elle vererek testler yapılmıştır ve k değerinin büyüdükçe algoritmanın performans ve doğruluk oranı değerlerinin düştüğü görülmüştür. Bu yüzden her iki yöntemin de en iyi performans ve doğruluk oranı değerlerini veren 1 k değeri olarak alınmıştır. K en yakın komşu algoritması k-CV yöntemi test sonuçları ve confusion matrix;

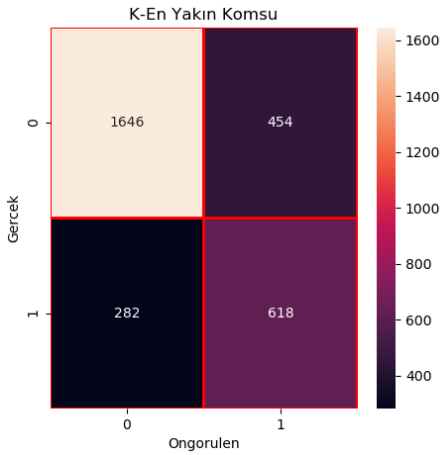
En iyi n komsu degeri (k) : 1

Başarılı veri adedi : 2264

KNN doğruluk oranı : 75%

KNN işlem süresi – sn : 0.51

KNN k-CV Değerleri : [0.72, 0.765, 0.77, 0.74, 0.74166667]



Şekil 19: K-Nearest Neighbor Hata Matrisi (k-CV Confusion Matrix)

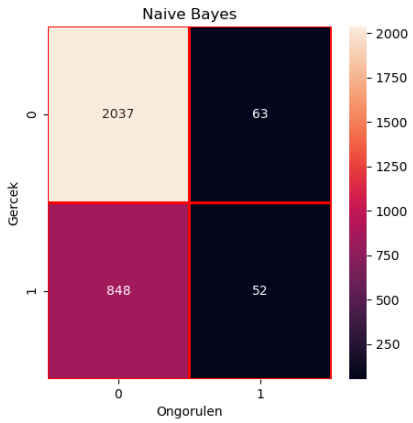
Naive Bayes algoritması k-CV yöntemi test sonuçları ve confusion matrix;

Başarılı veri adedi : 2089

NB doğruluk oranı : 70%

NB işlem süresi – sn : 0.04

NB k-CV Değerleri : [0.71333333, 0.68666667, 0.695, 0.70333333, 0.67666667]



Şekil 20: Naive Bayes Hata Matrisi (k-CV Confusion Matrix)

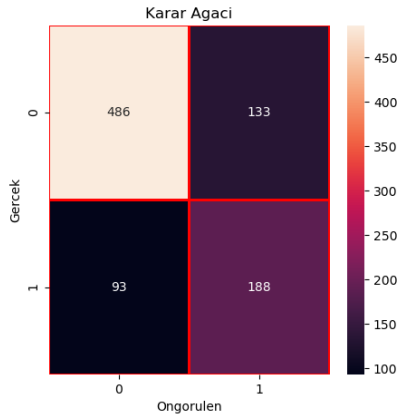
Karar Ağaçları algoritması k-CV yöntemi test sonuçları ve confusion matrix;

Başarılı veri adedi : 2267

DT doğruluk oranı : 76%

DT işlem süresi – sn: 0.17

DT k-CV Değerleri : [0.735, 0.76833333, 0.78666667, 0.75833333, 0.75833333]



Şekil 21: Decision Tree Hata Matrisi (k-CV Confusion Matrix)

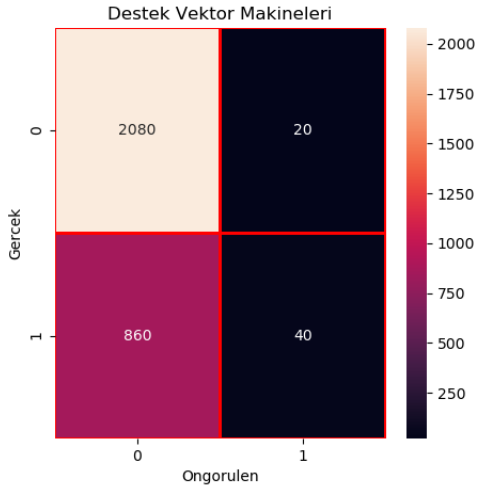
Destek Vektör Makineleri algoritması k-CV yöntemi test sonuçları ve confusion matrix;

Başarılı veri adedi : 2120

SVM doğruluk oranı : 71%

SVM işlem süresi – sn : 1.99

SVM k-CV Değerleri : [0.705, 0.71166667, 0.70166667, 0.705, 0.70666667]



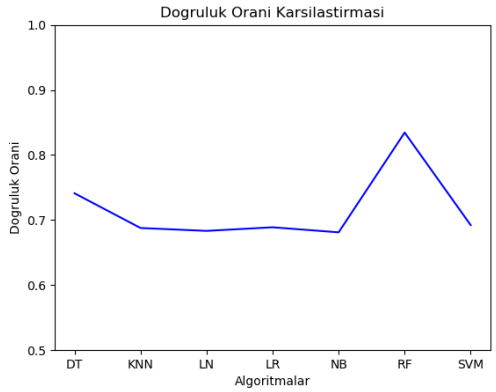
Şekil 22: Support Vector Machines Hata Matrisi (k-CV Confusion Matrix)

4. BULGULAR VE DEĞERLENDİRME

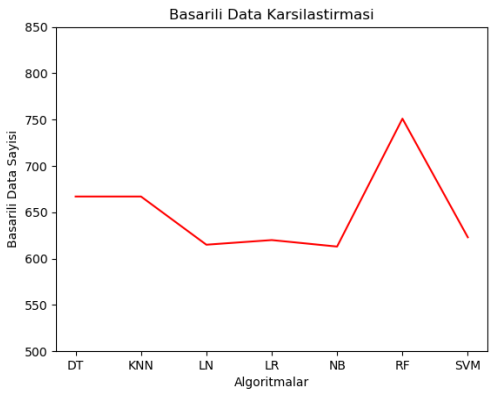
Sigorta sektöründe araç hasarları için hileli hasar tespiti ile ilgili yaptığım bu uygulamada 3000 kayıtlık gerçek veri seti üzerinden test train split ve k fold cross validation yöntemleri ile 7 adet algoritmayı karşılaştırdım. Uygulamada veri kümesi üzerinden fraud veya no fraud şeklinde iki sonuç alınmakta ve bunların doğruluk oranlarına bakılmaktadır. İlaveten algoritmaların işlem süreleri de hesaplanarak karşılaştırılmıştır. Test train spit yöntemine göre elde edilen sonuçlar tablo 2, şekil 23, 24 ve 25 üzerinde, k fold cross validation yöntemine göre elde edilen sonuçlar ise tablo 3, şekil 26, 27 ve 28 de gösterilmiştir.

Tablo 2: Bütün Algoritmalar İçin Alınan Sonuçlar (Test Train Split)

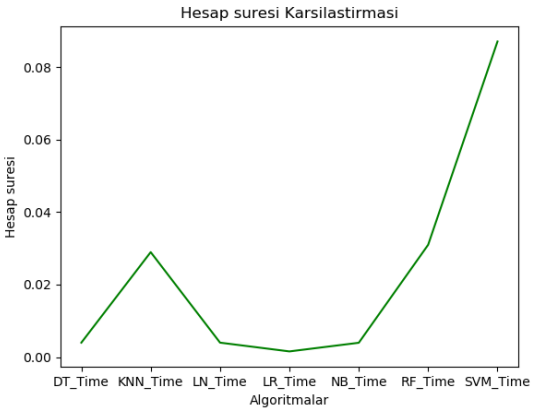
Model	Başarılı Data	Doğruluk	İşlem Süresi (sn)
Rasgele Orman	755	84%	0.041
Lojistik Regresyon	620	69%	0.002
Doğrusal Regresyon	615	68%	0.006
K En Yakın Komşular	667	69%	0.032
Naive Bayes	613	68%	0.003
Karar Ağaçları	664	74%	0.002
Destek Vektör Makineleri	623	69%	0.081



Şekil 23: Doğruluk Oranları



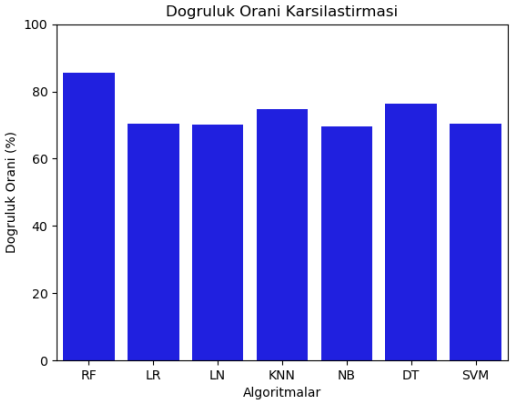
Şekil 24: Başarılı Data Oranları



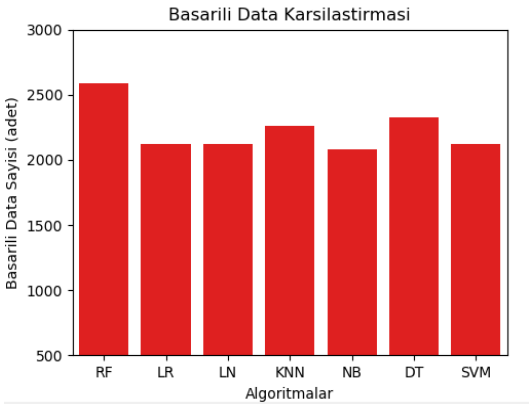
Şekil 25: İşlem Süreleri

Tablo 3: Tüm Algoritmaların Sonuçları (k Fold Cross Validation)

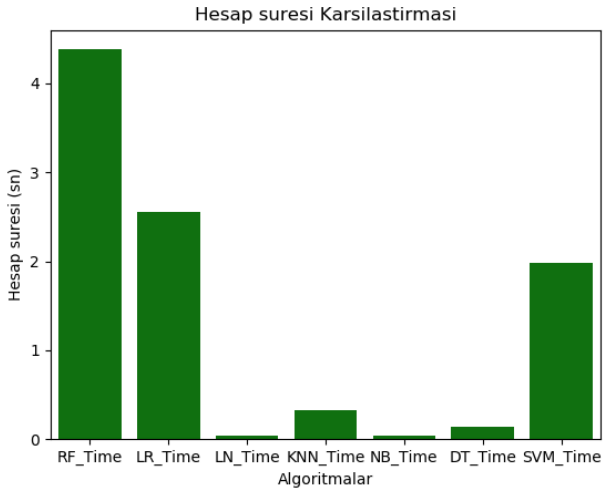
Model	Başarılı Data	Doğruluk	İşlem Süresi (sn)
Rasgele Orman	2578	86%	4.41
Lojistik Regresyon	2113	71%	2.41
Doğrusal Regresyon	2099	70%	0.08
K En Yakın Komşular	2264	75%	0.51
Naive Bayes	2089	70%	0.04
Karar Ağaçları	2267	76%	0.17
Destek Vektör Makineleri	2120	71%	1.99



Şekil 26: Doğruluk Oranları



Şekil 27: Başarılı Data Oranları



Şekil 28: İşlem Süreleri

Çıktılar incelendiğimizde her iki yöntemde de rasgele orman algoritmasının doğruluk oranlarının en yüksek olduğu görülmektedir. Test train split yöntemiyle %84, k fold cross validation yöntemiyle %86 oranında doğruluk oranı görülmektedir. Rasgele orman algoritmasını her iki yöntemde de karar ağaçlarının takip ettiği görülmüştür. Benzer şekilde %74 ve %76 oranlarında doğru tespit yapılmıştır. Bu iki algoritmayı k en yakın komşular algoritması takip etmektedir. K en yakın komşular algoritması k-CV yöntemiyle çalıştırıldığında doğruluk oranının bariz bir şekilde arttığı ve %75 oranında doğru tespit yaptığı saptanmıştır. Test train split yönteminde ilk iki algoritma yani rasgele orman ve karar ağaçları algoritmaları dışındakiler birbirine çok yakın değerlerde doğru tespit yaptığı görülmektedir. İşlem süreleri incelendiğinde rasgele orman algoritmasının doğruluk oranı yüksekliğinin aksine yavaş olduğu görülmektedir. İlk yöntemimiz olan test train split yönteminde en yavaş algoritma destek vektör makineleri olduğu görülüyor. Akabinde rasgele orman algoritması gelmektedir. Yine k-CV yöntemiyle yapılan testlerde de en yavaş algoritmanın rasgele orman olduğu görülmektedir. En hızlı çalışan algoritmanın her iki yöntem de baz alındığında naive bayes olduğu görülüyor. K-CV yönteminde naive bayes algoritmasını doğrusal regresyon takip etmektedir. Test train split yönteminde ise bir birine çok yakın değerler saptanmış ve lojistik, doğrusal regresyon, naive nayes ve karar ağaçları yüksek performans göstermiştir. Buradaki değerler veri setimizin büyüklüğüne bağlı olarak çok kısa süreler olarak saptanmış ve küçük farklar gösterse bile daha büyük veri setlerinin olduğu ortamlar için bize bir görüş vermektedir.

Karar Ağaçları ve Rasgele Orman algoritmaları denetimli birer makine öğrenimi algoritması olup, sınıflandırma tekniklerini kullanan algoritmalardır. Buradan yola çıkarak kullandığımız veri setimizin bu modellere ve algoritmalara daha uygun olduğu bilgisini vermektedir. En yüksek doğruluk oranı skorunu elde ettiğimiz rasgele orman algortiması ile varsayılan paramatre değerlerini değiştirerek daha yüksek bir skor elde etmek amacıyla yaptığımız çalışma sonuçlarında doğruluk oranı değerinin dikkate değer bir yükseliş göstermediği görülmüştür. Buradan yola çıkarak bütün algoritmaların testlerinde varsayılan değerleri kullanılarak uygulama çalıştırılmıştır. Sigorta sektörüne özel oto hasarlarında sahtecilik tespiti ile ilgili bu yönde çalışma sayılarının az olması yaptığım çalışmanın bu alanda çalışmalar yapmak isteyen uzman veya akademik çalışanlar için örnek teşkil edeceğini düşünmekteyim.

KAYNAKÇA

Alpaydın, E. (2020), Introduction to Machine Learning, no. 3.

Ayhan, S. ve Erdoğmuş, Ş. (2014), "Destek Vektör Makineleriyle Sınıflandırma Problemlerinin Çözümü İçin Çekirdek Fonksiyonu Seçimi", Eskişehir Osmangazi Üniversitesi İİBF Dergisi, 9(1), 175- 198

Barros R.C., de Carvalho A.C.P.L.F., Freitas A.A. (2015). Decision-Tree Induction. In: Automatic Design of Decision-Tree Induction Algorithms. No. 7-9. Doi: https://doi.org/10.1007/978-3-319-14231-9_2.

Bishop, M. (2006), Pattern Recognition and Machine Learning, no. 1.

BROWN, J.J. (1964) Life Insurance-Benefit or Fraud?, Toronto:Longmans Canada Limited.

Coalition Against Insurance Fraud (2006) Annual Report. USA: Coalition Against Insurance Fraud.

Demirci, S. (2019), Sigortacılıkta Yeni Bir Yaklaşım: Katılım Sigortacılığı, İnönü Üniversitesi Hukuk Fakültesi Dergisi – İnÜHFD 10(1): 25-39 (2019).

Derrig, R. A. (2002). Insurance fraud. Journal of Risk and Insurance, 69(3), 271–287. https://doi.org/10.1111/1539-6975.00026

Günbatar, E. (2019), Türkiye' de Otomobil Sigortası Sahtekarlıklarının Makine Öğrenmesi Yöntemleri ile Tespit Edilmesi, yüksek lisans tezi, Hacettepe Üniversitesi Endüstri Mühendisliği Anabilim Dalı, Ankara.

GÜVEL, Enver Alper; GÜVEL, Afitap ÖNDAS. (2008) Sigortacılık, 4. baskı, Ankara : Seçkin Yayıncılık.

Hazım, L.R. (2018), Four Classification Methods Naive Bayesian, Support Vector Machine, K-Nearest Neighbors and Random Forest Are Tested For Credit Card Fraud Detection, yüksek lisans tezi, Altınbaş Üniversitesi Fen Bilimleri Enstitüsü, İstnabul.

Joshi, A. V. (2019), Machine Learning and Artificial Intelligence, no. 10-11, 34, 37, https://dx.doi.org/10.1007/978-3-030-26622-6.

Kızılkaya, Y.M. ve Oğuzlar A. (2018), Bazı Denetimli Öğrenme Algoritmalarının R Programlama Dili ile Kıyaslanması, Karadeniz, 2018; (37).

LOUGHRAN, David S. "Deterring Fraud: The Role of General Damage Awards in Automobile Insurance Settlements", Journal of Risk and Insurance, Vol.72 No.4, ss.551-575.

Mengi, B. T. (2014). Araç Sigortası Hileleri ve Bu Hilelere Yönelik Önlemler. Finansal Araştırmalar ve Çalışmalar Dergisi, 4(8), 71–86. Retrieved from http://dergipark.gov.tr/marufacd/issue/502/4567

Narin, A, İşler, Y, Özer, M. (2014). Konjestif Kalp Yetmezliği Teşhisinde Kullanılan Çapraz Doğrulama Yöntemlerinin Sınıflandırıcı Performanslarının Belirlenmesine Olan Etkilerinin Karşılaştırılması. Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi, 16 (48), 1-8. [Çevrimiçi]. https://dergipark.org.tr/en/pub/deumffmd/issue/40797/492155 [Erişildi: 02.05.2020].

NILSSON, N.J. (1996) Introducion To Machine Learning, no. 1-2.

Oshiro T.M., Perez P.S., Baranauskas J.A. (2012) How Many Trees in a Random Forest?. Department of Computer Science and Mathematics Faculty of Philosophy, Sciences and Languages at Ribeirao Preto University, Sao Paulo. Doi: https://doi.org/10.1007/978-3-642-31537-4_13

ÖZBOLAT, Murat. (2009) Temel Sigortacılık, Seçkin Yayıncılık, Ankara.

Yıldırım, İ. (2013), Türk Sigortacılık Sektörünün Yumuşak Karnı: Sigorta Suistimalleri Sorunu, Sosyal ve Beşeri Bilimler Dergisi Cilt 5, No 1, 2013 ISSN: 1309-8012 (Online).

İnternet Kaynakları

http://www.wisegeek.com, [Çevrimiçi]. Available: http://www.wisegeek.com/what-is-life-insurance-fraud.htm [Erişildi: 01.05.2020]

https://siseb.sbm.org.tr, [Çevrimiçi]. Available: https://siseb.sbm.org.tr/tr/sisbis [Erişildi: 28.06.2020].

https://siseb.sbm.org.tr, [Çevrimiçi]. Available: https://siseb.sbm.org.tr/tr/istatistikler# [Erişildi: 28.06.2020].

https://www.businesswire.com, [Çevrimiçi]. Available: https://www.businesswire.com/news/home/20191120005771/en [Erişildi: 02.07.2020]

www.ibm.com, [Çevrimiçi]. Available: https://www.ibm.com/tr-tr/analytics/learn/linear-regression. [Erişildi: 08 03 2020].

www.insurancefraud.org, [Çevrimiçi]. Available: http://www.insurancefraud.org/scam-alerts-workers-compensation.htm#.UP7v2idFXuQ [Erişildi: 05.05.2020]

www.quackwatch.com, [Çevrimiçi]. Available: http://www.quackwatch.com/02ConsumerProtection/insfraud.html [Erişildi: 24.04.2020].

www.sas.com, [Çevrimiçi]. Available: https://www.sas.com/tr_tr/customers/sigorta-bilgi-merkezi.html. [Erişildi: 24.04.2020].

www.tsb.org.tr, [Çevrimiçi]. Available: https://www.tsb.org.tr/turkiyede-sigortacilik.aspx?pageID=439 [Erişildi: 29.06.2020]

www.turkishtimedergi.com, [Çevrimiçi]. Available: http://www.turkishtimedergi.com/sigorta/sigorta-sektorunde-sahtecilik-buyuk-veriyle-sucustu. [Erişildi: 24.04.2020].

EKLER

Test Train Split Modeli Kaynak Kodları

```
#ilgili moduller yukleniyor
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns # Goruntuleme (Heatmap icin)
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import accuracy_score
import warnings
from sklearn.metrics import confusion_matrix # Karsilastirma
from sklearn.model_selection import learning_curve
from sklearn.model_selection import ShuffleSplit
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from datetime import datetime
import time

#ignore Warnings
#warnings.filterwarnings("ignore")

#dataset import ediliyor
```

```
data=pd.read_csv('vals3.csv', sep='|')

#data baslik ve ornek kontrol
#print(data.columns)
#print(data.head())

data=data.drop(columns=['DOSYA_NO']) #istenmeyen kolon cikariliyor
data.columns

#test train data belirleme
y=data[data.columns[data.columns.isin(['FRAUD'])]].values.ravel()
X=data[data.columns[~data.columns.isin(['FRAUD'])]].values
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.30,
        random_state=42)



compare = []
time = []



def plot_learning_curve(estimator, title, X, y, axes=None, ylim=None, cv=None,
                        n_jobs=None, train_sizes=np.linspace(.1, 1.0, 5)):

    if axes is None:
        _, axes = plt.subplots(1, 3, figsize=(20, 5))

    axes.set_title(title)
    if ylim is not None:
        axes.set_ylim(*ylim)
    axes.set_xlabel("Egitim Ornekleri")
    axes.set_ylabel("Dogruluk Orani")
```

```
train_sizes, train_scores, test_scores, fit_times, _ = \
    learning_curve(estimator, X, y, cv=cv, n_jobs=n_jobs,
                   train_sizes=train_sizes,
                   return_times=True)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)
fit_times_mean = np.mean(fit_times, axis=1)
fit_times_std = np.std(fit_times, axis=1)

# Plot learning curve
axes.grid()
axes.fill_between(train_sizes, train_scores_mean - train_scores_std,
                   train_scores_mean + train_scores_std, alpha=0.1,
                   color="r")
axes.fill_between(train_sizes, test_scores_mean - test_scores_std,
                   test_scores_mean + test_scores_std, alpha=0.1,
                   color="g")
axes.plot(train_sizes, train_scores_mean, 'o-', color="r",
          label="Egitim Dogruluk Orani")
axes.plot(train_sizes, test_scores_mean, 'o-', color="g",
          label="Test Dogruluk Orani")
axes.legend(loc="best")

# # Plot n_samples vs fit_times
# axes[1].grid()
# axes[1].plot(train_sizes, fit_times_mean, 'o-')
# axes[1].fill_between(train_sizes, fit_times_mean - fit_times_std,
#                      fit_times_mean + fit_times_std, alpha=0.1)
# axes[1].set_xlabel("Egitim Ornekleri")
# axes[1].set_ylabel("fit_times")
```

```
    # axes[1].set_title("Modelin Olceklenebilirligi")

    # # Plot fit_time vs score
    # axes[2].grid()
    # axes[2].plot(fit_times_mean, test_scores_mean, 'o-')
    # axes[2].fill_between(fit_times_mean, test_scores_mean - test_scores_std,
    #                      test_scores_mean + test_scores_std, alpha=0.1)
    # axes[2].set_xlabel("fit_times")
    # axes[2].set_ylabel("Oran")
    # axes[2].set_title("Modelin Performansi")
    plt.tight_layout()

    return plt



#randomforest ####################
rf=RandomForestClassifier()
rf.fit(X_train, y_train)

#RANDOMFOREST paremetreleri..
# RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
#                        max_depth=None, max_features='auto', max_leaf_nodes=None,
#                        min_impurity_decrease=0.0, min_impurity_split=None,
#                        min_samples_leaf=1, min_samples_split=2,
#                        min_weight_fraction_leaf=0.0, n_estimators=10,
#                        n_jobs=None, oob_score=False, random_state=None,
#                        verbose=0, warm_start=False)
now = datetime.now()
pred=rf.predict(X_test)
rf_acc=accuracy_score(y_test,pred)
rf_cm = confusion_matrix(y_test,pred)
```

```
later = datetime.now()
t_rf = (later - now).total_seconds()
#print('RandomForest dogruluk:',rf_acc)
#print(rf_cm)

f,ax = plt.subplots(figsize = (5,5))
sns.heatmap(rf_cm, annot = True, linewidths = 1, linecolor = "red", fmt = ".0f", ax =
      ax)
plt.xlabel("Ongorulen")
plt.ylabel("Gercek")
plt.title("RandomForest")
plt.show()

rf_correct = rf_cm[1][1] + rf_cm[0][0]
print("Basarili data sayisi : ",rf_correct)
print("RandomForest dogruluk orani :", rf_acc)
print("RF Test Suresi :", t_rf)
print("RF CV Dizin:", rf_acc)

compare.append(["RF",rf_correct,rf_acc])
time.append(["RF_Time",t_rf])



#logisticRegression ile tahmin############
lr=LogisticRegression(max_iter=3000)
lr.fit(X_train,y_train)
#LgistcRegression parametreleri
# LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
#                intercept_scaling=1, l1_ratio=None, max_iter=100,
#                multi_class='warn', n_jobs=None, penalty='l2',
#                random_state=None, solver='warn', tol=0.0001, verbose=0,
#                warm_start=False)
```

```
now = datetime.now()
pred=lr.predict(X_test)
lr_acc=accuracy_score(y_test,pred)
lr_cm = confusion_matrix(y_test,pred)
later = datetime.now()
t_lr = (later - now).total_seconds()
#print('sonuc:',lr_acc,lr_cm,y_test,pred)



#print('LogisticRegression dogruluk:',lr_acc)
#print(lr_cm)


f,ax = plt.subplots(figsize = (5,5))
sns.heatmap(lr_cm, annot = True, linewidths = 1, linecolor = "red", fmt = ".0f", ax =
    ax)
plt.xlabel("Ongorulen")
plt.ylabel("Gercek")
plt.title("LogisticRegression")
plt.show()


lr_correct = lr_cm[1][1] + lr_cm[0][0]
print("Basarili data sayisi : ",lr_correct)
print("LR dogruluk orani :", lr_acc)
print("LR hesap suresi :", t_lr)


compare.append(["LR",lr_correct,lr_acc])
time.append(["LR_Time",t_lr])


#LinearRegression ile tahmin##########
ln=LinearRegression()
ln.fit(X_train,y_train)
#LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
    normalize=False) parametreleri
```

```
now = datetime.now()
pred_tmp=ln.predict(X_test)
pred = np.where(pred_tmp<=0.49,0,1)
ln_acc=accuracy_score(y_test,pred)
ln_cm = confusion_matrix(y_test,pred)
later = datetime.now()
t_ln = (later - now).total_seconds()
#print('LinearRegression dogruluk:',ln_acc)
#print(ln_cm)

f,ax = plt.subplots(figsize = (5,5))
sns.heatmap(ln_cm, annot = True, linewidths = 1, linecolor = "red", fmt = ".0f", ax =
      ax)
plt.xlabel("Ongorulen")
plt.ylabel("Gercek")
plt.title("LinearRegression")
plt.show()



ln_correct = ln_cm[1][1] + ln_cm[0][0]
#ln_accuracy = ln.score(X_test,y_test)
print("Basarili data sayisi : ",ln_correct)
print("LN dogruluk orani :", ln_acc)
print("LN Hesap suresi :", t_ln)

compare.append(["LN",ln_correct,ln_acc])
time.append(["LN_Time",t_ln])



#K en yakin komsu ##############
from sklearn.neighbors import KNeighborsClassifier
scores = []
```

```
for i in range(1,len(X_test)):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    scores.append(knn.score(X_test,y_test))



#Indexler 0 dan basladigi icin 1 fazlasi ekrana gosteriliyor
k_value = scores.index(max(scores))+1
print("En iyi n komsu degeri :", k_value)

knn2 = KNeighborsClassifier(n_neighbors=k_value)

#KNN parametrelerin ekrana yazilmasi
knn2.fit(X_train,y_train)
now = datetime.now()
y_predict = knn2.predict(X_test)
knn_cm = confusion_matrix(y_test,y_predict)
later = datetime.now()
t_knn = (later - now).total_seconds()
#cv = ShuffleSplit(n_splits=100, test_size=0.2, random_state=0)
#fig, axes = plt.subplots(1, figsize=(5,5))
#plot_learning_curve(knn2, "K-En Yakin Komsu", X, y,axes=axes, ylim=(0.2,
    1.01),cv=cv, n_jobs=4)
#plt.show()



f,ax = plt.subplots(figsize = (5,5))
sns.heatmap(knn_cm, annot = True, linewidths = 1, linecolor = "red", fmt = ".0f", ax
    = ax)
plt.xlabel("Ongorulen")
plt.ylabel("Gercek")
plt.title("K-En Yakın Komsu")
plt.show()
```

```
knn_correct = knn_cm[1][1] + knn_cm[0][0]
knn_acc = knn.score(X_test,y_test)
print("Basarili data sayisi : ",knn_correct)
print("KNN dogruluk orani : ", knn_acc)
print("KNN hesap suresi : ", t_knn)
compare.append(["KNN",knn_correct,knn_acc])
time.append(["KNN_Time",t_knn])


#Naive Bayes
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train,y_train)
now = datetime.now()
y_predict = nb.predict(X_test)
nb_cm = confusion_matrix(y_test,y_predict)
later = datetime.now()
t_nb = (later - now).total_seconds()


f,ax = plt.subplots(figsize = (5,5))
sns.heatmap(nb_cm, annot = True, linewidths = 1, linecolor = "red", fmt = ".0f", ax =
      ax)
plt.xlabel("Ongorulen")
plt.ylabel("Gercek")
plt.title("Naive Bayes")
plt.show()

nb_correct = nb_cm[1][1] + nb_cm[0][0]
nb_acc = nb.score(X_test,y_test)
print("Basarili data sayisi : ",nb_correct)
```

```
print("NB dogruluk orani : ", nb_acc)
print("NB hesap suresi : ", t_nb)
compare.append(["NB",nb_correct,nb_acc])
time.append(["NB_Time",t_nb])

#Decision Tree
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train,y_train) #parametrelerin ekrana yazilmasi
now = datetime.now()
y_predict = dt.predict(X_test)
dt_cm = confusion_matrix(y_test,y_predict)
later = datetime.now()
t_dt = (later - now).total_seconds()

#cv = ShuffleSplit(n_splits=100, test_size=0.2, random_state=0)
#fig, axes = plt.subplots(1, figsize=(5,5))
#plot_learning_curve(dt, "Karar Agaci", x, y,axes=axes, ylim=(0.7, 1.01),cv=cv,
      n_jobs=4)
#plt.show()

f,ax = plt.subplots(figsize = (5,5))
sns.heatmap(dt_cm, annot = True, linewidths = 1, linecolor = "red", fmt = ".0f", ax =
      ax)
plt.xlabel("Ongorulen")
plt.ylabel("Gercek")
plt.title("Karar Agaci")
plt.show()

dt_correct = dt_cm[1][1] + dt_cm[0][0]
dt_acc = dt.score(X_test,y_test)
print("Basarili data sayisi : ",dt_correct)
print("DT dogruluk orani : ", dt_acc)
```

```
print("DT hesap suresi : ", t_dt)
compare.append(["DT",dt_correct,dt_acc])
time.append(["DT_Time",t_dt])



#Support Vector#######################################
x_data = data.drop(["FRAUD"],axis = 1)
X=(x_data - np.min(x_data))/(np.max(x_data)-np.min(x_data))
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.30, random_state=42)


from sklearn.svm import SVC
svm = SVC(random_state = 42,C=2.0)
svm.fit(X_train,y_train) #parametrelerin ekrana yazilmasi
now = datetime.now()
y_predict = svm.predict(X_test)
svm_cm = confusion_matrix(y_test,y_predict)
later = datetime.now()
t_svm = (later - now).total_seconds()
#cv = ShuffleSplit(n_splits=100, test_size=0.2, random_state=0)
#fig, axes = plt.subplots(1, figsize=(5,5))
#plot_learning_curve(svm, "Destek Vektor Makineleri", x, y,axes=axes, ylim=(0.7, 1.01),cv=cv, n_jobs=4)
#plt.show()


f,ax = plt.subplots(figsize = (5,5))
sns.heatmap(svm_cm, annot = True, linewidths = 1, linecolor = "red", fmt = ".0f", ax = ax)
plt.xlabel("Ongorulen")
plt.ylabel("Gercek")
plt.title("Destek Vektor Makineleri")
plt.show()
```

```
svm_correct = svm_cm[1][1] + svm_cm[0][0]
svm_acc = svm.score(X_test,y_test)
print("Basarili data sayisi : ",svm_correct)
print("SVM dogruluk orani : ", svm_acc)
print("SV hesap suresi : ", t_svm)
compare.append(["SVM",svm_correct,svm_acc])
time.append(["SVM_Time",t_svm])

#########

accuracy = []
correct = []
index = []
for i in compare:
    accuracy.append(i[2])
    correct.append(i[1])
    index.append(i[0])
data = {"Correct":correct,"Accuracy":accuracy}

pd.options.display.float_format = '{:,.3f}'.format
df = pd.DataFrame(data,index = index)
print(df)

#sns.lineplot(index,correct,color = "red")
sns.barplot(index,correct,color = "red")
plt.ylim(500,850)
plt.xlabel("Algoritmalar")
plt.ylabel("Basarili Data Sayisi (adet)")
plt.title("Basarili Data Karsilastirmasi")
plt.show()

#sns.lineplot(index,accuracy,color = "blue")
```

```
sns.barplot(index,accuracy,color = "blue")
plt.ylim(0.5,1)
plt.xlabel("Algoritmalar")
plt.ylabel("Dogruluk Orani (%)")
plt.title("Dogruluk Orani Karsilastirmasi")
plt.show()



calculate = []
index = []
for i in time:
    calculate.append(i[1])
    index.append(i[0])
data = {"Calculate":calculate}

pd.options.display.float_format = '{:,.3f}'.format
df = pd.DataFrame(data,index = index)
print(df)

#sns.lineplot(index,calculate,color = "green")
sns.barplot(index,calculate,color = "green")
#plt.ylim(0.0001,0.05)
plt.xlabel("Algoritmalar")
plt.ylabel("Hesap suresi (sn)")
plt.title("Hesap suresi Karsilastirmasi")
plt.show()
```

K-Fold Cross Validation Yöntemi Kaynak Kodları:

```
#ilgili moduller yukleniyor
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns # Goruntuleme (Heatmap icin)
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import accuracy_score
import warnings
from sklearn.metrics import confusion_matrix # Karsilastirma
from sklearn.model_selection import learning_curve
from sklearn.model_selection import ShuffleSplit
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import StratifiedKFold
from datetime import datetime
import time

#ignore Warnings
#warnings.filterwarnings("ignore")

#dataset import ediliyor
data=pd.read_csv('vals3.csv', sep='|')

#data baslik ve ornek kontrol
#print(data.columns)
#print(data.head())
```

```
data=data.drop(columns=['DOSYA_NO']) #istenmeyen kolon cikariliyor
data.columns


'''
data['MODEL_YASI'].plot.hist() #model yas grafik
plt.title("Model Yas Grafik")
plt.xlabel("Araç Yaşı")
plt.ylabel("Hasar Adet")
plt.show()
data[data['FRAUD'] == 0 ]['MODEL_YASI'].plot.hist() #model yasa gore fraud hasar
    olmama durumu
plt.title("Model Yaşa Göre Gerçek Hasarlar")
plt.xlabel("Araç Yaşı")
plt.ylabel("Hasar Adet")
plt.show()
data[data['FRAUD'] == 1 ]['MODEL_YASI'].plot.hist() #model yasa gore fraud
    durumu
plt.title("Model Yaşa Göre Sahte Hasarlar")
plt.xlabel("Araç Yaşı")
plt.ylabel("Hasar Adet")
plt.show()
data[data['FRAUD'] == 1 ]['TEMINAT_BEDEL'].plot.hist() #arac bedele gore fraud
    durumu
plt.title("Model Yaşa Göre Sahte Hasarlar")
plt.xlabel("Araç Bedeli (x10bin TL)")
plt.ylabel("Hasar Adet")
plt.show()
'''


#test train data belirleme
y=data[data.columns[data.columns.isin(['FRAUD'])]].values.ravel()
X=data[data.columns[~data.columns.isin(['FRAUD'])]].values
```

```
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.30,
        random_state=42)



compare = []
time = []



def plot_learning_curve(estimator, title, X, y, axes=None, ylim=None, cv=None,
                        n_jobs=None, train_sizes=np.linspace(.1, 1.0, 5)):
    """
    Generate 3 plots: the test and training learning curve, the training
    samples vs fit times curve, the fit times vs score curve.

    Parameters
    ----------
    estimator : object type that implements the "fit" and "predict" methods
        An object of that type which is cloned for each validation.

    title : string
        Title for the chart.

    X : array-like, shape (n_samples, n_features)
        Training vector, where n_samples is the number of samples and
        n_features is the number of features.

    y : array-like, shape (n_samples) or (n_samples, n_features), optional
        Target relative to X for classification or regression;
        None for unsupervised learning.

    axes : array of 3 axes, optional (default=None)
        Axes to use for plotting the curves.
```

ylim : tuple, shape (ymin, ymax), optional
Defines minimum and maximum yvalues plotted.

cv : int, cross-validation generator or an iterable, optional
Determines the cross-validation splitting strategy.
Possible inputs for cv are:
- None, to use the default 5-fold cross-validation,
- integer, to specify the number of folds.
- :term:`CV splitter`,
- An iterable yielding (train, test) splits as arrays of indices.

For integer/None inputs, if ``y`` is binary or multiclass,
:class:`StratifiedKFold` used. If the estimator is not a classifier
or if ``y`` is neither binary nor multiclass, :class:`KFold` is used.

Refer :ref:`User Guide <cross_validation>` for the various
cross-validators that can be used here.

n_jobs : int or None, optional (default=None)
Number of jobs to run in parallel.
``None`` means 1 unless in a :obj:`joblib.parallel_backend` context.
``-1`` means using all processors. See :term:`Glossary <n_jobs>`
for more details.

train_sizes : array-like, shape (n_ticks,), dtype float or int
Relative or absolute numbers of training examples that will be used to
generate the learning curve. If the dtype is float, it is regarded as a
fraction of the maximum size of the training set (that is determined
by the selected validation method), i.e. it has to be within (0, 1].
Otherwise it is interpreted as absolute sizes of the training sets.
Note that for classification the number of samples usually have to

```
    be big enough to contain at least one sample from each class.
    (default: np.linspace(0.1, 1.0, 5))
    """
    if axes is None:
        _, axes = plt.subplots(1, 3, figsize=(20, 5))

    axes.set_title(title)
    if ylim is not None:
        axes.set_ylim(*ylim)
    axes.set_xlabel("Egitim Ornekleri")
    axes.set_ylabel("Dogruluk Orani")

    train_sizes, train_scores, test_scores, fit_times, _ = \
        learning_curve(estimator, X, y, cv=cv, n_jobs=n_jobs,
                       train_sizes=train_sizes,
                       return_times=True)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    fit_times_mean = np.mean(fit_times, axis=1)
    fit_times_std = np.std(fit_times, axis=1)

    # Plot learning curve
    axes.grid()
    axes.fill_between(train_sizes, train_scores_mean - train_scores_std,
                      train_scores_mean + train_scores_std, alpha=0.1,
                      color="r")
    axes.fill_between(train_sizes, test_scores_mean - test_scores_std,
                      test_scores_mean + test_scores_std, alpha=0.1,
                      color="g")
    axes.plot(train_sizes, train_scores_mean, 'o-', color="r",
```

```
            label="Egitim Dogruluk Orani")
    axes.plot(train_sizes, test_scores_mean, 'o-', color="g",
            label="Test Dogruluk Orani")
    axes.legend(loc="best")

    # # Plot n_samples vs fit_times
    # axes[1].grid()
    # axes[1].plot(train_sizes, fit_times_mean, 'o-')
    # axes[1].fill_between(train_sizes, fit_times_mean - fit_times_std,
    #                 fit_times_mean + fit_times_std, alpha=0.1)
    # axes[1].set_xlabel("Egitim Ornekleri")
    # axes[1].set_ylabel("fit_times")
    # axes[1].set_title("Modelin Olceklenebilirligi")

    # # Plot fit_time vs score
    # axes[2].grid()
    # axes[2].plot(fit_times_mean, test_scores_mean, 'o-')
    # axes[2].fill_between(fit_times_mean, test_scores_mean - test_scores_std,
    #                 test_scores_mean + test_scores_std, alpha=0.1)
    # axes[2].set_xlabel("fit_times")
    # axes[2].set_ylabel("Oran")
    # axes[2].set_title("Modelin Performansi")
    plt.tight_layout()

    return plt



#randomforest ####################
rf=RandomForestClassifier()
#rf.fit(X_train, y_train)
```

```
#RANDOMFOREST paremetreleri..
# RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
#                        max_depth=None, max_features='auto', max_leaf_nodes=None,
#                        min_impurity_decrease=0.0, min_impurity_split=None,
#                        min_samples_leaf=1, min_samples_split=2,
#                        min_weight_fraction_leaf=0.0, n_estimators=10,
#                        n_jobs=None, oob_score=False, random_state=None,
#                        verbose=0, warm_start=False)
now = datetime.now()
#pred=rf.predict(X_test)
skf = StratifiedKFold(n_splits=5,shuffle=True)
#rf_acc=accuracy_score(y_test,pred)
y_pred = cross_val_predict(rf, X, y, cv=skf)
#rf_cm = confusion_matrix(y_test,pred)
rf_cm = confusion_matrix(y,y_pred)
rf_acc = cross_val_score(rf, X, y, cv=skf)
later = datetime.now()
t_rf = (later - now).total_seconds()
#print('RandomForest dogruluk:',rf_acc)
#print(rf_cm)

f,ax = plt.subplots(figsize = (5,5))
sns.heatmap(rf_cm, annot = True, linewidths = 1, linecolor = "red", fmt = ".0f", ax =
      ax)
plt.xlabel("Ongorulen")
plt.ylabel("Gercek")
plt.title("RandomForest")
plt.show()

rf_correct = rf_cm[1][1] + rf_cm[0][0]
print("Basarili data sayisi : ",rf_correct)
print("RF Dogruluk Orani: %.2f%%" % (rf_acc.mean()*100.0))
```

```
print("RF Test Suresi :", t_rf)
print("RF CV Dizin:", rf_acc)
rf_acc2 =  (rf_acc.mean()*100.0)
compare.append(["RF",rf_correct,rf_acc2])
time.append(["RF_Time",t_rf])

'''
rf=RandomForestClassifier(n_estimators=20)  #n_estimators degeri arttırılıyor
rf.fit(X_train, y_train)
pred=rf.predict(X_test)
print('RandomForest 2:',accuracy_score(y_test,pred))

rf=RandomForestClassifier(min_samples_split=4,                    n_estimators=20)
      #min_samples_split degeri arttırılıyor..
rf.fit(X_train, y_train)
pred=rf.predict(X_test)
print('RandomForest 3:',accuracy_score(y_test,pred))
'''
#logisticRegression ile tahmin############
lr=LogisticRegression(max_iter=3000)
#lr.fit(X_train,y_train)
#LgistcRegression parametreleri
# LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
#                intercept_scaling=1, l1_ratio=None, max_iter=100,
#                multi_class='warn', n_jobs=None, penalty='l2',
#                random_state=None, solver='warn', tol=0.0001, verbose=0,
#                warm_start=False)
now = datetime.now()
#pred=lr.predict(X_test)
skf = StratifiedKFold(n_splits=5,shuffle=True)
#lr_acc=accuracy_score(y_test,pred)
lr_acc = cross_val_score(lr, X, y, cv=skf)
```

```
y_pred = cross_val_predict(lr, X, y, cv=skf)
#lr_cm = confusion_matrix(y_test,pred)
lr_cm = confusion_matrix(y, y_pred)
later = datetime.now()
t_lr = (later - now).total_seconds()
#print('sonuc:',lr_acc,lr_cm,y_test,pred)



#print('LogisticRegression dogruluk:',lr_acc)
#print(lr_cm)


'''
cv = ShuffleSplit(n_splits=100, test_size=0.2, random_state=0)
fig, axes = plt.subplots(1, figsize=(5,5))
plot_learning_curve(lr, "Lojistik Regresyon", X, y,axes=axes, ylim=(0.2, 1.01),cv=cv,
        n_jobs=4)
plt.show()
'''
f,ax = plt.subplots(figsize = (5,5))
sns.heatmap(lr_cm, annot = True, linewidths = 1, linecolor = "red", fmt = ".0f", ax =
        ax)
plt.xlabel("Ongorulen")
plt.ylabel("Gercek")
plt.title("LogisticRegression")
plt.show()


lr_correct = lr_cm[1][1] + lr_cm[0][0]
print("Basarili data sayisi : ",lr_correct)
print("LR Dogruluk Orani: %.2f%%" % (lr_acc.mean()*100.0))
print("LR hesap suresi :", t_lr)
print("LR CV Dizin", lr_acc)
lr_acc2 = (lr_acc.mean()*100.0)
compare.append(["LR",lr_correct,lr_acc2])
```

```
time.append(["LR_Time",t_lr])



#LinearRegression ile tahmin##########
ln=LinearRegression()
#ln.fit(X_train,y_train)
#LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
    normalize=False) parametreleri
now = datetime.now()
skf = StratifiedKFold(n_splits=5,shuffle=True)
pred_tmp=cross_val_predict(ln, X, y, cv=skf)
y_pred = np.where(pred_tmp<=0.49,0,1)
ln_cm = confusion_matrix(y,y_pred)
ln_acc=cross_val_score(ln, X, y, cv=skf)
later = datetime.now()
t_ln = (later - now).total_seconds()
#print('LinearRegression dogruluk:',ln_acc)
#print(ln_cm)


f,ax = plt.subplots(figsize = (5,5))
sns.heatmap(ln_cm, annot = True, linewidths = 1, linecolor = "red", fmt = ".0f", ax =
    ax)
plt.xlabel("Ongorulen")
plt.ylabel("Gercek")
plt.title("LinearRegression")
plt.show()



ln_correct = ln_cm[1][1] + ln_cm[0][0]
#ln_accuracy = ln.score(X_test,y_test)
print("Basarili data sayisi : ",ln_correct)
print("LN Dogruluk Orani: %.2f%%" % (ln_acc.mean()*100.0))
print("LN Hesap suresi :", t_ln)
```

```
print("LN CV Dizin :", ln_acc)
#ln_acc2 = (ln_acc.mean()*100.0)
ln_acc2 = 70.002
compare.append(["LN",ln_correct,ln_acc2])
time.append(["LN_Time",t_ln])


#K en yakin komsu ###############
from sklearn.neighbors import KNeighborsClassifier
'''
scores = []
skf = StratifiedKFold(n_splits=5,shuffle=True)
for i in range(1,len(X)):
    knn = KNeighborsClassifier(n_neighbors=i)
    #knn.fit(X_train,y_train)
    knn_acc2 = (cross_val_score(knn, X, y, cv=skf))
    scores.append(knn_acc2.mean())
#Indexler 0 dan basladigi icin 1 fazlasi ekrana gosteriliyor
k_value = scores.index(max(scores))+1
print("En iyi n komsu degeri :", k_value)
'''
knn2 = KNeighborsClassifier(n_neighbors=1)
#KNN parametrelerin ekrana yazilmasi
#knn2.fit(X_train,y_train)
now = datetime.now()
skf = StratifiedKFold(n_splits=5,shuffle=True)
y_predict = cross_val_predict(knn2, X, y, cv=skf)
knn_cm = confusion_matrix(y,y_predict)
knn_acc = cross_val_score(knn2, X, y, cv=skf)
later = datetime.now()
t_knn = (later - now).total_seconds()
#cv = ShuffleSplit(n_splits=100, test_size=0.2, random_state=0)
#fig, axes = plt.subplots(1, figsize=(5,5))
```

```
#plot_learning_curve(knn2, "K-En Yakin Komsu", X, y,axes=axes, ylim=(0.2, 1.01),cv=cv, n_jobs=4)
#plt.show()



f,ax = plt.subplots(figsize = (5,5))
sns.heatmap(knn_cm, annot = True, linewidths = 1, linecolor = "red", fmt = ".0f", ax = ax)
plt.xlabel("Ongorulen")
plt.ylabel("Gercek")
plt.title("K-En Yakın Komsu")
plt.show()

knn_correct = knn_cm[1][1] + knn_cm[0][0]
print("Basarili data sayisi : ",knn_correct)
print("KNN Dogruluk Orani: %.2f%%" % (knn_acc.mean()*100.0))
print("KNN hesap suresi : ", t_knn)
print("KNN CV Dizin :", knn_acc)
knn_acc2 = (knn_acc.mean()*100.0)
compare.append(["KNN",knn_correct,knn_acc2])
time.append(["KNN_Time",t_knn])



#Naive Bayes
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
now = datetime.now()
skf = StratifiedKFold(n_splits=5,shuffle=True)
#nb.fit(X_train,y_train)
#y_predict = nb.predict(X_test)
y_predict = cross_val_predict(nb, X, y, cv=skf)
nb_cm = confusion_matrix(y,y_predict)
nb_acc = cross_val_score(nb, X, y, cv=skf)
```

```
later = datetime.now()
t_nb = (later - now).total_seconds()
'''
cv = ShuffleSplit(n_splits=100, test_size=0.2, random_state=0)
fig, axes = plt.subplots(1, figsize=(5,5))
plot_learning_curve(nb, "Naive Bayes", x, y,axes=axes, ylim=(0.7, 1.01),cv=cv, n_jobs=4)
plt.show()
'''


f,ax = plt.subplots(figsize = (5,5))
sns.heatmap(nb_cm, annot = True, linewidths = 1, linecolor = "red", fmt = ".0f", ax = ax)
plt.xlabel("Ongorulen")
plt.ylabel("Gercek")
plt.title("Naive Bayes")
plt.show()

nb_correct = nb_cm[1][1] + nb_cm[0][0]
print("Basarili data sayisi : ",nb_correct)
print("NB Dogruluk Orani: %.2f%%" % (nb_acc.mean()*100.0))
print("NB hesap suresi : ", t_nb)
print("NB CV Dizin :", nb_acc)
nb_acc2 = (nb_acc.mean()*100.0)
compare.append(["NB",nb_correct,nb_acc2])
time.append(["NB_Time",t_nb])

#Decision Tree
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
now = datetime.now()
skf = StratifiedKFold(n_splits=5,shuffle=True)
#dt.fit(X_train,y_train) #parametrelerin ekrana yazilmasi
```

```
y_predict = cross_val_predict(dt, X, y, cv=skf)
dt_cm = confusion_matrix(y,y_predict)
dt_acc = cross_val_score(dt, X, y, cv=skf)
later = datetime.now()
t_dt = (later - now).total_seconds()

#cv = ShuffleSplit(n_splits=100, test_size=0.2, random_state=0)
#fig, axes = plt.subplots(1, figsize=(5,5))
#plot_learning_curve(dt, "Karar Agaci", x, y,axes=axes, ylim=(0.7, 1.01),cv=cv,
    n_jobs=4)
#plt.show()

f,ax = plt.subplots(figsize = (5,5))
sns.heatmap(dt_cm, annot = True, linewidths = 1, linecolor = "red", fmt = ".0f", ax =
    ax)
plt.xlabel("Ongorulen")
plt.ylabel("Gercek")
plt.title("Karar Agaci")
plt.show()

dt_correct = dt_cm[1][1] + dt_cm[0][0]
print("Basarili data sayisi : ",dt_correct)
print("DT Dogruluk Orani : %.2f%%" % (dt_acc.mean()*100.0))
print("DT hesap suresi : ", t_dt)
print("DT CV Dizin :", dt_acc)
dt_acc2 = (dt_acc.mean()*100.0)
compare.append(["DT",dt_correct,dt_acc2])
time.append(["DT_Time",t_dt])

#Support Vector#######################################
x_data = data.drop(["FRAUD"],axis = 1)
X=(x_data - np.min(x_data))/(np.max(x_data)-np.min(x_data))
```

```
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.30, random_state=42)


from sklearn.svm import SVC
svm = SVC(random_state = 42,C=2.0)
now = datetime.now()
skf = StratifiedKFold(n_splits=5,shuffle=True)
#svm.fit(X_train,y_train) #parametrelerin ekrana yazilmasi
y_predict = cross_val_predict(svm, X, y, cv=skf)
svm_cm = confusion_matrix(y,y_predict)
svm_acc = cross_val_score(svm, X, y, cv=skf)
later = datetime.now()
t_svm = (later - now).total_seconds()
#cv = ShuffleSplit(n_splits=100, test_size=0.2, random_state=0)
#fig, axes = plt.subplots(1, figsize=(5,5))
#plot_learning_curve(svm, "Destek Vektor Makineleri", x, y,axes=axes, ylim=(0.7, 1.01),cv=cv, n_jobs=4)
#plt.show()


f,ax = plt.subplots(figsize = (5,5))
sns.heatmap(svm_cm, annot = True, linewidths = 1, linecolor = "red", fmt = ".0f", ax = ax)
plt.xlabel("Ongorulen")
plt.ylabel("Gercek")
plt.title("Destek Vektor Makineleri")
plt.show()


svm_correct = svm_cm[1][1] + svm_cm[0][0]
print("Basarili data sayisi : ",svm_correct)
print("Accuracy: %.2f%%" % (svm_acc.mean()*100.0))
print("SV hesap suresi : ", t_svm)
print("SVM CV Dizin :", svm_acc)
svm_acc2 = (svm_acc.mean()*100.0)
```

```
compare.append(["SVM",svm_correct,svm_acc2])
time.append(["SVM_Time",t_svm])

#########

accuracy = []
correct = []
index = []
for i in compare:
    accuracy.append(i[2])
    correct.append(i[1])
    index.append(i[0])
data = {"Correct":correct,"Accuracy":accuracy}

pd.options.display.float_format = '{:,.3f}'.format
df = pd.DataFrame(data,index = index)
print(df)

#sns.lineplot(index,correct,color = "red")
sns.barplot(index,correct,color = "red")
plt.ylim(500,3000)
plt.xlabel("Algoritmalar")
plt.ylabel("Basarili Data Sayisi (adet)")
plt.title("Basarili Data Karsilastirmasi")
plt.show()

#sns.lineplot(index,accuracy,color = "blue")
sns.barplot(index,accuracy,color = "blue")
plt.ylim(0,100)
plt.xlabel("Algoritmalar")
plt.ylabel("Dogruluk Orani (%)")
plt.title("Dogruluk Orani Karsilastirmasi")
```

```
plt.show()



calculate = []
index = []
for i in time:
    calculate.append(i[1])
    index.append(i[0])
data = {"Calculate":calculate}

pd.options.display.float_format = '{:,.3f}'.format
df = pd.DataFrame(data,index = index)
print(df)

#sns.lineplot(index,calculate,color = "green")
sns.barplot(index,calculate,color = "green")
#plt.ylim(0.0001,0.05)
plt.xlabel("Algoritmalar")
plt.ylabel("Hesap suresi (sn)")
plt.title("Hesap suresi Karsilastirmasi")
plt.show()
```

ÖZGEÇMİŞ

Ad-Soyad : Yaşar GEREN

Doğum Tarihi ve Yeri : 20.09.1979 / İSTANBUL

E-posta : yasargeren@gmail.com

ÖĞRENİM DURUMU:

Lisans : Anadolu Üniversitesi – İktisat

Ön Lisans : Ömer Halis Demir Üniversitesi – Bilgisayar Programcılığı 1998

MESLEKİ DENEYİM :

2019 - ... : Ağ ve Bilgi Sistemleri Güvenliği - Yönetici / AXA Sigorta.

1999 - 2019 : Sistem ve Network Yönetimi - Uzman / AXA Sigorta

1998 – 1999 : BT Sistem Uzmanı / Aras Kargo

1997 : Teknik Servis Stajyeri / Perkom Bilgi İşlem