Büyük Veri İçeren Öneri Sistemleri İçin Hiperparametre Optimizasyonu

Merve Pınar[1] , Orhan Okumuş[1] , Umut Orçun Turgut[2] , Oya Kalıpsız[1] ,

Mehmet S. Aktaş[1]

[1] Bilgisayar Mühendisliği Bölümü, Elektrik-Elektronik Fakültesi
Yıldız Teknik Üniversitesi, İstanbul
[2] Ar-Ge Merkezi, Cybersoft, İstanbul
merve.pinar@std.yildiz.edu.tr, orhan.okumus@std.yildiz.edu.tr,
umut.turgut@cybersoft.com.tr, kalipsiz@yildiz.edu.tr,
aktas@yildiz.edu.tr

Özet. Finans sektöründe müşteriye doğru bir öneri sunmak en önemli noktalardan biridir. Öneri sistemlerinde yüksek başarımlı bir veri madenciliği uygulaması gerçekleştirirken belirlenen temel algoritmaların her birinin birçok parametre girdisi bulunmaktadır. Veri madenciliği alanında uzman bir kişi kullanacağı veri kümesine göre bahsedilen parametre değerlerini rahatlıkla tanımlayabilmektedir. Bu çalışmada, veri madenciliği alanında daha deneyimsiz kişilere kolay bir şekilde uygun parametre değerlerinin sunulması ve bu parametre değerlerine göre sonuçların çıkartılması amaçlanmıştır. Geliştirilen yazılımda, büyük veri kümeleri üzerinde hızlı ve paralel işlem yapmaya olanak sağlayan Apache Spark'a ait MLlib kütüphanesindeki algoritmalardan yararlanılarak hiperparametre optimizasyonu iki farklı yaklaşımla uygulanmıştır. İlk yaklaşımda seçilen veri kümesi üzerinde uygulanmak istenen algoritma seçimini yapan kullanıcıdan parametreler için sınır değeri istenirken, ikinci yaklaşımda bu sınır değerlerine gerek kalmadan optimizasyon işlemi yapılmaktadır. Her iki yaklaşımda da denetimli ve denetimsiz öğrenme algoritmaları bankacılık alanında yer alan gerçek veri kümeleri üzerinde test edilmiş ve sonuçlar karşılaştırılmıştır.

Anahtar Kelimeler: Hiperparametre Optimizasyonu, Veri Madenciliği, Denetimli ve Denetimsiz Öğrenme Algoritmaları.

Hyperparameter Optimization for Recommendation Systems with Big Data

Merve Pınar[1] , Orhan Okumuş[1] , Umut Orçun Turgut[2] , Oya Kalıpsız[1] ,

Mehmet S. Aktaş[1]

[1] Computer Engineering Department, Electrical and Electronics Faculty
Yıldız Technical University, İstanbul
[2] R&D Center, Cybersoft, İstanbul
merve.pinar@std.yildiz.edu.tr, orhan.okumus@std.yildiz.edu.tr,
umut.turgut@cybersoft.com.tr, kalipsiz@yildiz.edu.tr,
aktas@yildiz.edu.tr

Abstract. One of the most important points in finance sector is to recommend the right products to the customer. Each of the basic algorithms determined when implementing a high performance data mining application in the recommendation systems has many parameters. An expert in the field of data mining can easily describe the parameter values according to the data set that will use. In this paper, it is aimed to present the appropriate parameter values of the more inexperienced users in the field of data mining and to draw a conclusion according to these parameter values. In the developed software, hyperparameter optimization has been implemented with two different approaches by using the algorithms in the Apache Spark MLlib library, which allows fast and parallel processing on large data sets. In the first approach, when the user chooses the algorithm to be applied on the selected data set, the boundaries are required for the parameters. In the second approach, the optimization process is performed without the necessity of these boundaries. In both approaches, supervised and unsupervised learning algorithms were tested on a real life banking data sets and the results were compared.

Keywords: Hyperparameter Optimization, Data Mining, Supervised and Unsupervised Algorithms

1 Giriş

Gelişen veri depolama kapasitesiyle birlikte bankaların veri tabanları zenginleşmiştir. Artan müşteri sayısı ve buna bağlı artan işlem kapasitesiyle müşteri ihtiyaçlarını hızlı ve doğru bir şekilde tespit ederek çözüm önerileri sunmak daha da zor bir hale gelmiştir. Bu sorunun çözümü için yenilikçi veri madenciliği uygulamaları ve tekniklerine ihtiyaç duyulmaktadır.

Makine öğrenmesi, eğitim verisini girdi alarak veri tabanlı tahminleri ve kararları gerçekleştirmek amacıyla model inşa etmeye odaklanmaktadır. Eğitim işleminden önce kullanılacak her bir denetimli veya denetimsiz öğrenme algoritmasının başlatılabilmesi için parametre değerlerine ihtiyaç duyulması makine öğrenmesinde yaygınlaşmış bir problemdir. Bu parametreler hiperparametre olarak adlandırılır ve oluşturulan model üzerinde çok etkili değişimlere sebep olabilirler. Hiperparametreler, verilen bir öğrenme algoritmasına ait öğrenme oranı, kernel parametreleri, ağ mimarisi gibi pek çok sonucu yapılandırmak için kullanılır.

Model oluşturulurken başarı oranı yüksek sonuçlar alabilmek için hiperparametrelerin optimize edilmiş bir şekilde verilmesi gerekmektedir. Veri madenciliği alanında uzman bir kullanıcı uygun parametre değerini kolay bir şekilde hesaplayabilirken, uzmanlık gerektirmeyecek bir şekilde başka bir kullanıcı geliştirilen yazılım sayesinde uygun değerlerini öğrenebilecek ve bu değerlere göre işlemini gerçekleştirecektir. Bu çalışmada optimizasyon iki farklı yaklaşımla ele alınmaktadır. Birinci yaklaşımda kullanıcı çalıştıracağı algoritmanın seçimine ek olarak bu algoritmaya ait parametrelerin sınır değerleri de belirleyebilecek seviyede olmalıdır. İkinci yaklaşımda ise kullanıcı sadece veri kümesi ve algoritma seçimini yaparak sonuçları alabilmektedir.

Yazılım gerçekleştirilirken, denetimli ve denetimsiz öğrenme için iki yönteme ait temel algoritmalar seçilmiştir. İşlem yapılacak veri kümesinin çalışacağı algoritmaya uygun olması kullanıcı tarafından beklenen uzmanlık seviyesine dahildir. Her bir algoritmanın kabul ettiği hiperparametre farklılığı ve bu parametrelerin sonsuz farklı değer alabilmesi problemin karmaşıklığını artırmaktadır. Bu geniş aralıklara kısıtlamalar getirerek model optimizasyonu kolaylaştırılmıştır. Bu yüzden, parametre değerleri altkümesini tarayan ızgara arama veya belirli sayıda deneme yapan Monte Carlo rastlantısal yaklaşım kullanıcının tercihine sunulmuştur.
Sistemin oluşturulmasında Apache Spark MLlib kütüphanesinden[1] yararlanılmıştır. Öneri sisteminin finans sektörü dahilinde bir bankaya ait olması sebebiyle güvenliğin sağlanması adına karartılmış bankacılık verileri ile test edilmiştir ve değerlendirme kriterlerine göre sonuçlar karşılaştırılmıştır.

Bu bildirinin yazım organizasyonu şu şekildedir; Bölüm 2'de; hiperparametre optimizasyonu konusunda yapılan benzer çalışmalar anlatılacaktır. Bölüm 3'te; belirlenen denetimli ve denetimsiz öğrenme algoritmalarının tanımlaması yapılacak ve bu algoritmalar üzerine optimizasyon işleminin nasıl gerçekleştirildiği açıklanacaktır. Bölüm 4'te; uygulama ve testler incelenecektir. Bölüm 5'te; başarılı bir şekilde sonlanan uygulamanın sonuçları incelenecek ve gelecek araştırmalar için tavsiyeler ele alınacaktır.

2 İlgili Çalışmalar

Ürün öneri sistemlerinin gereksinimlerine göre belirlenen birçok model bulunur ve her modelin birçok parametresi bulunmaktadır. Belirli bir modelin parametrelerini optimize etmek için günümüzde TUPAQ [2], Auto-Weka [3], HyperOpt [5] gibi çeşitli yazılımlar mevcuttur. Bahsedilen yazılımlar kullanmış oldukları kütüphanelerle ağırlıklı olarak sınıflandırma algoritmaları üzerine odaklanmışlardır.

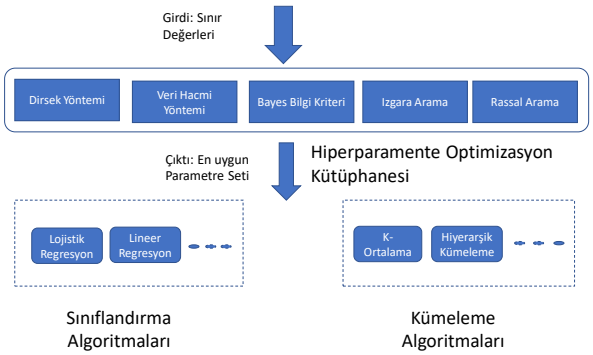
TUPAQ, MLbase sisteminin bir bileşenidir ve tahminsel uygulamalar için bulunanlarla karşılaştırılabilir nitelikte kapsamlı stratejileri kullanarak, standart yaklaşımlardan daha verimli bir şekilde modelleri bulan ve bu modelleri eğiten bir yazılımdır. Maliyet tabanlı kaynak tahsis etmeyi, geliştirilmiş hiperparametre optimizasyon tekniklerini içeren otomatik makine öğrenimi için bir mimari olarak önerilmiştir.

HyperOpt, Python'da model seçimi için algoritmaları ve paralel altyapıyı içermektedir. Kullanıcılara bir konfigürasyon alanını ve bu alan içerisindeki noktalara gerçek değerler atayan bir değerlendirme fonksiyonunu ayıran optimizasyon ara yüzü sağlamaktadır. Kullanıcılardan olasılık dağılımıyla belirtilmesi istenen konfigürasyon alanı sayesinde uzmanların uygun değerlere karar vermesini kolaylaştırır.

Auto-Weka, Bayesian optimizasyonundaki [4] son yeniliklerden yararlanarak özellik seçimi ve değerlendirme yöntemlerinde geniş bir aralığa sahip olmasıyla sınıflandırma yaklaşımlarında başarılı bir performans göstermektedir. Eş zamanlı olarak makine öğrenmesi algoritmasının seçimini ve hiperparametre ayarlamalarını ayrı ayrı ele alarak incelemektedir.

3 Metodoloji

Geliştirilen yazılımda, model oluşturmak için veri kümesi üzerinde çalışacak algoritmalar denetimli ve denetimsiz öğrenme algoritmaları olarak iki başlık altında toplanmıştır. Denetimli öğrenme algoritmaları için lojistik regresyon, lineer regresyon, rastgele orman(random forest); denetimsiz öğrenme algoritmaları için k-ortalamalar (k-means), hiyerarşik kümeleme, beklenti-maksimizasyon yöntemleri seçilmiştir. Seçilen algoritmalarla ilgili kısa açıklamalar Bölüm 3.1 ve Bölüm 3.2' de yer almaktadır. Sistemin genel mimarisi Şekil 1' de yer almaktadır.



Şekil 1. Sistemin Genel Mimarisi

3.1. Çalışmada Kullanılan Kümeleme Algoritmaları

K-ortalamalar kümeleme(K-Means) [6], en yaygın denetimsiz öğrenme yöntemlerinden biridir ve her verinin sadece bir kümeye ait olmasına izin vermektedir. Amaç, gerçekleştirilen bölümleme işlemi sonunda elde edilen kümelerin, küme içi benzerliklerinin maksimum ve kümeler arası benzerliklerinin minimum olmasını sağlamaktır.

Hiyerarşik kümeleme [7], veri kümesindeki özgün grupları belirlemek için kullanılır. Uygulama sonucunda gözlemler dendogram olarak adlandırılan bir ağaç yapısı içerisinde sergilenir ve kümeleme kriteri olarak ikili bir uzaklık matrisi kullanılmaktadır.

Beklenti-maksimizasyon [8], tam olmayan veri problemlerini çözmek için maksimum olasılık tahminlerini yapan tekrarlı bir algoritmadır ve her tekrarı iki adımda gerçekleşir. Bu adımlar, bekleneni bulma ve maksimizasyon olarak adlandırılır.

3.2. Çalışmada Kullanılan Sınıflandırma Algoritmaları

Lojistik regresyon [9], verilerin bir lojistik fonksiyona uymasıyla bir olayın oluşma ihtimalini öngören bir regresyondur. Birçok regresyon analizi biçimi gibi, sayısal veya kategorik olabilen birkaç öngörme değişkeninden faydalanır.

Lineer regresyon [10], aralarında sebep-sonuç ilişkisi bulunan iki veya daha fazla değişken arasındaki ilişkiyi belirlemek ve bu ilişkiyi kullanarak o konu ile ilgili tahminler ya da kestirimler yapabilmek amacıyla yapılır. Sıklıkla tercih edilen yöntemlerden biridir.

Rastgele Orman(Random Forest) [11] , bir sınıflandırıcı yerine birden çok sınıflandırıcı üreten ve sonrasında onların tahminlerinden alınan oylar ile yeni veriyi sınıflandıran öğrenme algoritmasıdır ve kaybolan verilerin tahmin edilmesinde etkili bir metottur.

Çalışmamızda, kullanıcı birinci yaklaşımı tercih ederse seçtiği algoritmanın ihtiyaç duyduğu parametreler için sınır değerlerini girmesi beklenmektedir; ikinci yaklaşım tercih edilirse sınıflandırma algoritmaları için parametreler bir parametre haritasına yerleştirilir, ızgara [12] veya Monte Carlo rassal aramadan [13] biri seçilerek optimi-

zasyon işlemi yürütülmektedir. İkinci yaklaşımda kümeleme algoritmaları için dirsek yöntemi, veri hacmi yöntemi ve beklenti-maksimizasyon algoritmasına özel olarak bayes bilgi kriteri yöntemi ile optimizasyon işlemi yürütülmektedir. Bu yöntemlerin çalışmamızda nasıl kullanıldığı Bölüm 3.3 ve Bölüm 3.4' te detaylı olarak anlatılmaktadır.

3.3. Sınıflandırma Algoritmaları İçin Hiperparametre Optimizasyonu

Literatürde denetimli öğrenme algoritmaları olarak geçen sınıflandırma algoritmaları için gerçekleştirdiğimiz optimizasyon dört adımdan oluşmaktadır:

- Veri kümesi eğitim ve test verisi olmak üzere iki parçaya bölünür.
- Model eğitimi için ayrılan veriye seçili sınıflandırma algoritması oluşturulmuş parametre haritalarındaki parametrelerle iteratif şekilde uygulanır. Bu şekilde modeller üretilir.
- Üretilen modeller seçili sınıflandırma algoritmasına uygun değerlendirici fonksiyonu ile değerlendirilir.
- Veri kümesi için en iyi performans göstermiş parametre kombinasyonuna sahip model seçilir.

Bu çalışmada yararlanılan Apache Spark MLlib kütüphanesi hiperparametre optimizasyonu için Cross-Validation ve Train Validation Split olmak üzere 2 adet araç sunmaktadır.

Cross validation, model oluştururken tüm veriyi kullanmayı sağlayan bir metottur. Veriyi k (genellikle 10) adet sayıda parçaya böler. Bir parçasını test kalanını model eğitimi için kullanır. Eğitim için ayırdığı veriden model üretip test verisi üzerinde deneyerek en uygun modeli seçme odaklıdır. Bu işlemi test için ayrılan veri seti her seferinde farklı olacak şekilde k defa model seçilir. Seçilenler arasından en iyisi alınır. K adet iterasyon olduğu için maliyeti yüksektir buna karşı hata oranı düşüktür.

Train Validation metodunun Cross Validation'dan farkı veriyi bir kere bölmesidir. Verilen orana göre veriyi eğitim ve test veri kümelerine böler. Bu işlemi bir kere gerçekleştirdiğinden dolayı Cross Validation'a göre maliyeti azdır fakat veri setinin yeterince büyük olmadığı durumlarda daha az güvenilirdir. Cross-Validation ve Train Validation metotları aşağıdaki ögeleri girdi olarak almaktadır.

- Tahminleyici: Spark MLlib bünyesinde sınıflandırıcı ve regresyon algoritmalarını kendi araçlarına hizmet edebilir halde tasarlamış olup Tahminleyici adı altında toplamıştır. Bu öğe kullanıcının veri setine uygulamak istediği algoritmadır.

- Parametre Haritası: Tahminleyici olarak kullanılacak algoritmaya ait optimizasyonu istenen parametreleri ve değerlerini barındıran veri yapısıdır.
- Değerlendirici: Üretilen modellerin test verisine uygunluğunu ölçen metriktir. MLlib sınıflandırma ve regresyon işlemleri için iki çeşit değerlendirici sunmaktadır.

3.4. Kümeleme Algoritmaları İçin Hiperparametre Optimizasyonu

Literatürde denetimsiz öğrenme algoritmaları olarak geçen kümeleme yöntemleri veriye ait bir sınıf etiketi tanımlamamalarından dolayı bir önceki aşamadaki gibi Spark MLlib'in sunmuş olduğu optimizasyon araçlarına dahil edilememektedir. Bu nedenle kümeleme yöntemlerine hiperparametre optimizasyonu yapmak için 2 farklı sezgisel yöntem kullanmaktayız. Bunlar otomatikleştirilmiş dirsek yöntemi ve veri sesi yöntemi olarak ifade edilmektedir. Ayrıca beklenti maksimizasyonu algoritması üzerinde bu algoritmanın sonucunda elde edilen "loglikelihood" değerini kullanarak hesaplama yapan Bayes Bilgi Kriteri yöntemi uygulanmıştır.

Dirsek yönteminde (Bknz. Şekil 2) [13] k adet küme merkezi ile yürütülmesi tamamlanmış algoritma sonucunda, veriye ait karesel hata oranı değerleri bir grafik eğrisi oluşturularak değerlendirilmektedir. Amaç eğride dirsek şekline benzeyen, keskin bir düşüşün yaşandığı ve devamında stabil şekilde azalan noktayı tespit etmektir. Bu nokta veri kümesi için optimum küme merkezi sayısı olarak seçilmektedir.

Algoritma 1: Dirsek Yöntemi

1. k=1
2. Başla
3. k=k+1
4. En uygun çözümün "hataların karesi toplamı"nı hesapla
5. Önemli ölçüde düşüş gösteren k değeri optimum nokta
6. Bitir

Şekil 2. Dirsek yöntemine ait sözde kod

Veri hacmi yönteminde [14], verideki n adet boyuta ait "hacim" adında bir değer tanımlanmıştır. İki boyutlu bir yapıda "hacim" bir kümenin elemanlarını içine alabilen en küçük dikdörtgen alan olarak tanımlanabilir. Yöntem işleyiş olarak kümenin "hacim " değerini hesaplama ve optimum küme merkezini bulma olarak iki parçalı şekilde sürdürülmektedir (Bknz. Şekil 3 ve Şekil 4).

Algoritma 2: Hacim Değerini Hesapla

1. Başla
2. Her bir küme için Adım 2-4 tekrar et
3. Veri kümesi içindeki en büyük değeri bul
4. Veri kümesi içindeki en küçük değeri bul
5. O küme için en büyük değerden en küçük değeri çıkar ve "aralık" değerine eşitle
6. "Hacim" değerini o küme için tüm aralıkların toplamına eşitle
7. Bitir

Şekil 3. Veri hacmi yöntemine ait sözde kod

Algoritma 3: Optimum Küme Merkezini Bulma

1. Başla
2. Bir küme ile başla (k=1)
3. Bu küme için hacim değerlerini hesapla ve hacim dizisinde sakla
4. k=k+1
5. Yeni k değeri için K-Ortalamalar Kümeleme çalıştır
6. Oluşan tüm kümeler için hacim değerlerini hesapla ve hacim dizisinde sakla
7. Yeni hacim değerini bir önceki hacim değerine böl ve "oran" değerine eşitle
8. If oran < "sabit değer" then
9. Adım 3 tekrar et
10. Else
11. Kümelemeyi sonlandır ve bir sonraki adıma geç
12. Optimum nokta = k-1
13. Bitir

Şekil 4. Veri hacmi yöntemine ait sözde kod

Kullanıcı kümeleme işlemini kendi istediği bir parametre aralığında gerçekleştirmek isterse, birinci yaklaşım otomatikleştirilmiş dirsek yöntemini kullanarak tanımlı aralıktaki optimum parametre değerlerini ve kümeleme sonucunu kullanıcıya sunmaktadır. Eğer kullanıcı aralık vermek istemeyip veri kümesinin hangi parametre değerleriyle optimum sonucu vermek istediğini görmek isterse, ikinci yaklaşım, "hacim" bazlı kümeleme yöntemiyle kullanıcıya uygun bulunan parametre kombinasyonu ve sonuç kullanıcıya sunulmaktadır.

Bayes Bilgi Kriteri[15], beklenti-maksimizasyon algoritmasındaki "loglikelihood" değerini Formül 1'deki formüle yerleştirerek Bayes Bilgi Kriteri adında doğruluk değeri elde eder. En küçük çıkan değer için küme sayısı optimum seçilir.

Algoritma 4. Bayes Bilgi Kriterini Kullanarak Değerlendirme

1. Başla
2. Loglikelihood=0
3. Başlangıç değerleri için Beklenti-Maksimizasyon algoritması çalıştır
4. Elde edilen loglikelihod değerini Formül 3.1'e yerleştir
5. Formül sonucundaki BIC değerini dizide sakla
6. Belirlenen maksimum iterasyon sayısına kadar Adım 2-5 tekrar et
7. BIC dizisindeki en düşük değeri veren parametreler optimum değerler
8. Bitir

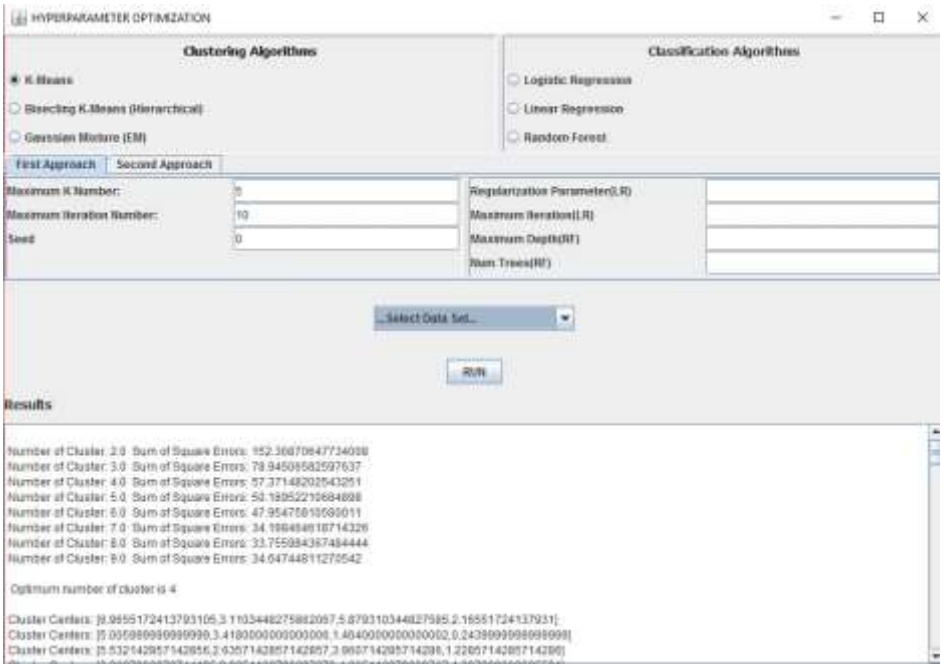
Şekil 5. Bayes Bilgi Kriteri yöntemine ait sözde kod

$$BIC(\lambda) = \sum_{i=1}^{n} \log\left\{\sum_{m=1}^{M} \pi_m \phi(x_i; \mu_m, \sum_m)\right\} - \frac{1}{2} MD_f \log n$$

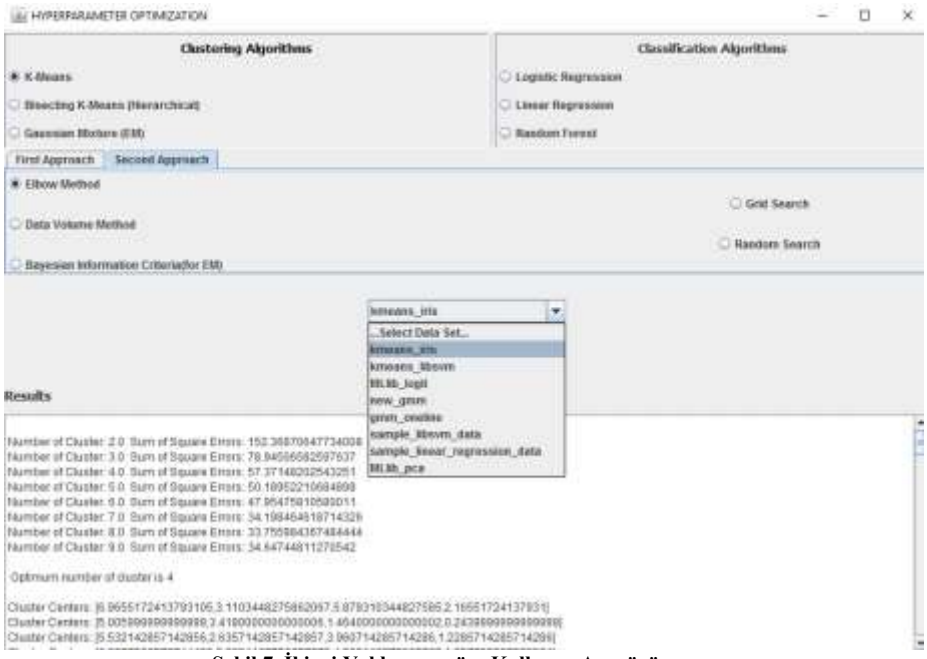
Formül 1. Bayes Bilgi Kriteri Formülü

4 Uygulama ve Test

Çalışmanın uygulaması Java programlama dili ile her iki yaklaşıma uygun olarak tasarlanmıştır. Uygulamada Java 1.8 ve Apache Spark 2.1.0 kullanılmıştır. Uygulamanın çalıştırılması için en az 2GB RAM ve 400MB Disk alanına ihtiyaç duyulmaktadır. Kullanıcı arayüzü Şekil 6 ve Şekil 7'de gösterilmiştir.



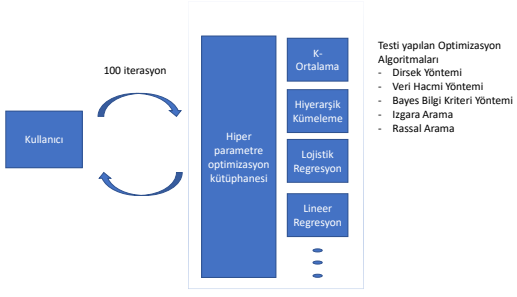
Şekil 6. Birinci Yaklaşıma göre Kullanıcı Arayüzü



Şekil 7. İkinci Yaklaşıma göre Kullanıcı Arayüzü

Uygulama arayüzü dört bölümden oluşmaktadır. İlk bölümde kümeleme ve sınıflandırma algoritmalarından birinin seçimi yapılır. İkinci bölümde birinci yaklaşım tercih edilecekse algoritmanın çalışması için gerekli parametrelerin sınır değerlerinin kullanıcı tarafından girilmesi beklenmektedir. Eğer ikinci yaklaşım tercih edilecekse kümeleme ve sınıflandırma algoritması üzerinde uygulanacak yöntemin kullanıcı tarafından seçilmesi beklenmektedir. Üçüncü bölümde seçilen algoritmaya uygun veri kümesinin seçimi yapılır ve uygulama çalıştırılır. Dördüncü ve son bölümde sonuçlar kullanıcıya sunulmaktadır.

Testler bu uygulama üzerinden Spark lokal modda çalıştırılarak Intel Core i7 model 2.20GHz işlemciye sahip, 4 çekirdekli bilgisayarda gerçekleştirilmiştir. Gerçekleştirilen test ortamı Şekil 8'de gösterilmiştir.



Şekil 8. Test Ortamı Tasarımı

Testler uygulamadaki tüm algoritmalar için gerçekleştrilmiştir. Sınıflandırma algoritmaları aynı veri kümesi(dosya boyutu 103KB) kullanılarak ızgara ve rassal arama için 100 defa çalıştırılmış ve sonuçların gösterilmesine kadar geçen süre ortalama ve standart sapma olarak Tablo 1'de sunulmuştur. Kümeleme algoritmaları aynı veri kümesi(dosya boyutu 5KB) kullanılarak dirsek, veri hacmi, bayes bilgi kriteri(beklenti-maksimizasyon algoritması için geçerli)(BBK) yöntemlerine göre 100 defa çalıştırılmış ve sonuçların gösterilmesine kadar geçen süre ortalama ve standart sapma olarak Tablo 2'de sunulmuştur. Çalışma süreleri hesaplanırken Java'da Timestamp sınıfı kullanılmış ve nanosaniye olarak elde edilmiştir. Uygulamanın çalışma süresi açısından maliyeti test edilmek için kullanılan veri kümesinin boyutuna göre değişmektedir.

Tablo 1. Kümeleme Algoritmaları Çalışma Süresi

Algoritmalar	Dirsek Yöntemi		Veri Hacmi Yöntemi		BBK Yöntemi	
	Ort. (sn)	Std. (sn)	Ort. (sn)	Std. (sn)	Ort. (sn)	Std. (sn)
K-Ortalama(K-Means)	4,14	0,55	3,36	0,67	---	---
Hiyerarşik Kümeleme	15,87	1,81	12,22	3,10	---	---
Beklenti-Maksimizasyon	---	---	2,89	1,17	7,11	1,05

Tablo 2. Sınıflandırma Algoritmaları Çalışma Süresi

Algoritmalar	Izgara Arama		Rassal Arama	
	Ort. (sn)	Std. (sn)	Ort. (sn)	Std. (sn)
Lojistik Regresyon	16,55	1,67	9,72	1,52
Lineer Regresyon	8,25	1,9	5,46	1,81
Rastgele Orman(Random Forest)	7,23	2,21	5,14	1,92

Çalışma süreleri karşılaştırıldığında seçilen yöntemlere göre belirgin farklar gözlemlenmiştir. Kümeleme algoritmaları için, her bir iterasyondaki kümelenen verilere göre kümeleme işlemine devam kararını vermesinden dolayı veri hacmi yöntemi dirsek yönteminden kısa sürede optimum parametre değerini vermiştir.

Sınıflandırma algoritmaları için belirlenen aralıktaki tüm değerleri sırayla deneyerek en uygun parametre değerini bulan ızgara arama yönteminin, bu aralıktaki değerleri sırayla denemek yerine aralarından seçim yaparak en uygun parametre değerini hesaplayan rassal arama yöntemine göre daha fazla zaman aldığı gözlemlenmiştir ve rassal aramada bütçe kriteri uygun parametreyi hesaplarken toplam iterasyon sayısı olarak belirlenmiştir. Bütçe dahilinde rassal arama veriye göre, parametre uzayına göre ve en önemli olarak zamana göre yeniden ölçeklendirilebilir.

5 Sonuçlar ve Gelecekteki Çalışmalar

Bu çalışma kapsamında, veri madenciliği alanında deneyimsiz kişilerin kullanmak istedikleri algoritmalarda girdi olarak verecekleri hiper parametre değerlerininin optimizasyonu sağlanarak uygun değerlere göre sonuç almaları için çözüm geliştirilmiştir. Belirlenen kümeleme ve sınıflandırma algoritmaları üzerinde süre açısından maliyeti az olan yöntemleri kararlaştırmak adına uygun parameter değerlerini hesaplamayı sağlayan farklı yöntemler uygulanmıştır.

Bölüm 3'te detaylı olarak açıklanmış yöntemlerden kümeleme algoritmaları için veri hacmi yöntemi tecrübenin az olup sınırların geniş olması durumlarında daha efektif çıkmıştır. Veri hacmi yönteminin diğer yöntemlerden öne geçtiği bir diğer sebep de kolaylıkla ölçeklenebilir olmasıdır. Sistemden beklenen azami başarı oranına sadece "eşik değeri(threshold)" üzerinden oynama yapılarak ulaşılabilir. Spesifik amaçlı veya veriye dair yeterli tecrübenin bulunduğu durumlarda da dirsek ve bayes bilgi kriteri ile değerlendirme metodları daha uygun bulunmuştur.

Sınıflandırma algoritmaları için ızgara araması, bir parametre alanını sistematik olarak araştırır ancak yalnızca sabit noktalara bakar. Bu bağlamda optimal noktaların, ızgara aramasında bakılan sabit noktalardizisinde bulunmaması tamamen mümkündür. Burada hangi algoritmanın daha uygun olduğu kullanım amacına göre değişmektedir. Eğer belirli yaygın parametre aralıklarında sonuç alınmak isteniyorsa ızgara araması verdiği sonuçların lineerliğinden dolayı daha uygun görülebilir. Aksine parametre uzayının geniş ve aranılan aralıkların belirli olmadığı durumlarda rassal arama tercih edilmelidir.

Uygulamanın doğru sonuçları vermesi için kullanılacak olan veri kümesinin ön işlemden geçirilmiş olması ve çalıştırılmak istenen algoritmaya göre uygun olması gerekmektedir.

Önerilen çözüm gelecekte seçilen veri kümesinin özelliklerine göre doğru sonuca daha az sürede ulaştıracak yöntemin kararlaştırılması ile iyileştirilebilir böylelikle kullanıcının optimum hiper parametre değerlerini elde etmesi için yapacağı işlemler azaltılabilir.

Teşekkür

Projemizi Ar-Ge ortamını kullanarak geliştirmemize sebep olan ve her türlü desteği biz lere sağlayan Cybersoft'a teşekkürlerimizi sunarız.

Kaynaklar

1. Spark MLlib Main Guide, https://spark.apache.org/docs/2.1.0/mllib-guide.html, son erişim 2017/03/25.
2. Michael J. Franklin, Evan R. Sparks, Tim Kraska: TuPAQ: An Efficient Planner for Large-scale Predictive Analytic Queries, Berkeley University, 2015.

3. James Bergstra, Dan Yamins, David D. Cox: Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms, SCIPY, 2013.
4. Thomas Huijskens: Bayesian optimization with scikit-learn, 29 Dec 2016.
5. Chris Thornton, Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown: Auto-Weka: Combined Selection and Hyperparameter Optimization of Classification Algorithms, University of British Columbia, 2012.
6. Güncel Sarıman: Veri Madenciliğinde Kümeleme Teknikleri Üzerine Bir Çalışma: K-Means ve K-Medoids Kümeleme Algoritmalarının Karşılaştırılması, süleyman Demirel Üniversitesi, 2011.
7. Hiyerarşik Kümeleme Analizi ve R İle Görselleştrilmesi, https://tr.linkedin.com/pulse/hiyerar%C5%9Fik-k%C3%BCmeleme-analizi-hc-ve-r-ile-datalab-tr, son erişim 2017/04/12.
8. Evren Sezgin, Yüksel Çelik: Veri Madenciliğinde Kayıp Veriler İçin Kullanılan Yöntemlerin Karşılaştırılması, Akdeniz Üniversitesi, 2013.
9. Lesson 6: Logistic Regression, https://onlinecourses.science.psu.edu/stat504/node/149, son erişim 2017/04/08.
10. Barış Şentuna: Regresyon Analizi, Balıkesir Üniversitesi, 2013.
11. Predrag Radenković: Random Forest, University Of Belgrade, 2015.
12. Tuning the hyper-parameters of an estimator, http://scikit-learn.org/stable/modules/grid_search.html , son erişim 2017/04/15.
13. Zelda B. Zabinsky: Random Search Algorithms, Washington University, 2009.
14. Purnima Bholowalia, Arvind Kumar: EBK-Means: A Clustering Technique based on Elbow Method and K-Means in WSN, Lovely Professional University, 2014.
15. Tao Huang, Heng Peng, Kun Zhang: Model Selection for Gaussian Mixture Models,Academia Sinica, 2013.
16. Abhijit Kane: Determining The Number of Clusters For A K-Means Clustering Algorithm, Birla Institute of Technology and Science, 2012.