

**İSTANBUL TECHNICAL UNIVERSITY
FACULTY OF COMPUTER AND
INFORMATICS**

ROSSMANN MAĞAZA SATIŞLARI TAHMİNİ

Bitirme Projesi

**Harun Çatal
150130034**

**Department: Computer Engineering
Division: Computer Engineering**

Advisor : Prof. Dr. Şule Gündüz Öğüdücü

Ocak 2019

Özgünlük Bildirisi

1. Bu çalışmada, başka kaynaklardan yapılan tüm alıntılar, ilgili kaynaklar referans gösterilerek açıkça belirtildiğini,
2. Alıntılar dışındaki bölümlerin, özellikle projenin ana konusunu oluşturan teorik çalışmaların ve yazılım/donanımın benim tarafımdan yapıldığını bildiririm.

İstanbul, 2019 Harun Çatal

ROSSMANN STORE SALES FORECASTING

(SUMMARY)

Rossmann is a company that consists of a chain of stores selling cosmetics and pharmaceutical products, especially in Europe. Rossmann has regularly recorded sales data for 1115 stores in Germany.

These data consist of many factors such as store sales, promotions, holidays, competition, periods. In the data set that we will use in the project, there is information about the affected factors as well as sales data.

The project was presented to users as a competition on the Kaggle [1] platform in 2015. The data set required for the project is available on Kaggle. The project concept includes data engineering, machine learning, feature derivation and regression algorithms. To achieve the result, many changes were made on the training data set and made ready for the algorithm. These changes can be counted as clearing the data, editing the properties, adding new features. The effects of the changes on the project results were examined.

Random Forest Regression and XGBoost regression were used as models in the algorithm section. The effects of the added features on these algorithms were measured and the necessary analyzes were performed.

The aim of the project is to create a sales forecast model by using this data set and to estimate the next six weeks' sales data for these stores. In this way, managers are informed in advance and necessary measures and investments are guided to increase internal efficiency.

As a result of the project, an RMSPE value of 0.146836 in the Random Forest Regression was approached with an XGBoost regression with an RMSPE of 0.172421.

ROSSMANN MAĞAZA SATIŞLARI TAHMİNİ

(ÖZET)

Rossmann, özellikle Avrupa'da yaygınlık gösteren ilaç ve kozmetik ürünleri satan mağazalar zincirinden oluşan bir şirkettir. Rossmann şirketi Almanya'da bulunan 1115 mağazasının düzenli olarak satış verilerini kayıt altına almıştır.

Bu veriler mağaza satışları, promosyonlar, tatiller, rekabet, dönemler gibi birçok faktörden oluşmaktadır. Projede kullanacağımız veri kümesinde satış verilerinin yanı sıra etkilenilen faktörler hakkında da bilgi mevcuttur.

Proje 2015 yılında Kaggle [1] platformunda bir yarışma olarak kullanıcılara sunulmuştu. Proje için gereken veri kümesi Kaggle üzerinde bulunmaktadır. Proje konsept olarak veri mühendisliği, makine öğrenmesi, özellik türetme ve regresyon algoritmalarını uygulama adımlarını kapsamaktadır. Sonuca ulaşmak için eğitim veri kümesi üzerinde birçok değişiklik yapılarak algoritmaya hazır hale getirildi. Bu değişiklikler veriyi temizlemek, özellikleri düzenlemek, anlamlandırmak, yeni özellikler eklemek olarak sayılabilir. Yapılan değişikliklerin proje sonucuna olan etkileri incelendi.

Algoritma bölümünde Random Forest Regresyonu ve XGBoost regresyonu model olarak kullanıldı. Eklenen özelliklerin bu algoritmalar üzerindeki etkileri ölçüldü, gerekli analizler yapıldı.

Projenin amacı elimizdeki bu veri setini kullanıp bir satış tahmin modeli oluşturarak bu mağazalar için gelecek altı haftalık satış verilerini tahmin etmektir. Bu sayede yöneticiler önceden bilgilendirilip gerekli önlemlerin ve yatırımların yönlendirilmesi sağlanarak şirket içi verimin artırılması hedeflenmektedir.

Projenin sonucu olarak Random Forest Regresyonunda 0.146836 lik bir RMSPE değeri, XGBoost regresyonuyla 0.172421' lik bir RMSPE değeri ile sonuca yaklaşmış olundu.

İçindekiler Tablosu

1	GİRİŞ	2
2	LİTERATÜR TARAMASI	3
2.1	Random Forest Regresyonu	3
2.2	XGBoost Regresyonu	3
2.3	NumPy	4
2.4	Pandas	4
2.5	Sklearn	5
2.6	IPython Notebook	5
2.7	Spyder	5
2.8	Matplotlib	5
3	ANALİZ VE MODELLEME	6
3.1	Veri Seti Analizi	6
3.2	Veri Analizi	8
3.3	Proje Diyagramı	12
4	PROJENİN UYGULANMASI	13
4.1	Verinin Temizlenmesi	13
4.1.1	NaN verilerin düzenlenmesi	13
4.1.2	Açık ve satış yapan mağazaların alınması	13
4.1.3	Tarih özelliğini parçalama	13
4.1.4	Haritalama işlemleri	14
4.1.5	Süre aralığının anlamlandırılması	14
4.1.6	Tarihin promosyon aralığında olduğunun kontrolü	15
4.2	Ekstra Özellik Türetme	15
4.3	Regresyonların Uygulanması	17
4.3.1	Random Forest Regresyonu	17
4.3.2	XGBoost Regresyonu	18
5	SONUÇ VE DEĞERLENDİRME	19
5.1	Random Forest Regresyonunun Sonucu	19
5.2	XGBoost Regresyonunun Sonucu	20
5.3	Sonucun Görselleştirilmesi	21
6	SONUÇ	24
7	KAYNAKÇA	25

1 GİRİŞ

Satış performansını tahmin etmek, her işletmenin karşı karşıya olduğu temel zorluklardan biridir. Firmaların, doğru ürünü doğru zamanda ve doğru yerde sunmaları için müşteri taleplerini öngörmesi önemlidir. Büyük firmalar için bu konu daha da önemli hale gelmektedir. Firmalar bu verilere göre hareket politikalarını belirlerler. Hangi mağazaya daha çok önem vermeleri gerektiğini saptayabilir, stok verilerini buna göre düzenleyebilir ve daha birçok önlem alabilir.

Rossmann şirketi Avrupa'nın çeşitli ülkelerinde yayılmış olan 3000'den fazla mağazası olan kozmetik ve ilaç ticareti yapan bir şirkettir. Üzerinde çalıştığımız proje Rossmann şirketinin satış müdürlerine verdiği bir görevi içermektedir. Görevin amacı 1 Ocak 2013 ile 31 Temmuz 2015 tarihleri arasındaki 1115 Rossmann mağazasına ait satış verilerini kullanarak 856 Rossmann mağazasının 17 Eylül 2015' e kadar olan satışlarını öngörmektir. Proje Kaggle [1] sitesinde bir yarışma olarak topluma sunuldu.

Mağaza satışları şüphesiz birçok faktörden etkilenmektedir. Bunlar arasında promosyonlar, rekabet, hafta içi veya hafta sonu olması durumu, okul tatili ve resmî tatiller, mevsimsellik, yöresellik, hava durumu gibi faktörler vardır. En önemli noktalardan birisi veriyi olabildiğince temizleyip, anlamlı hale getirmektir. Veri matematiksel olmayan ifadelerden oluşabilmektedir. Bizim amacımız verideki bu anlamsız ifadeleri matematikleştirerek modelin anlayabileceği duruma getirmektir. Projede bu faktörler veya daha sonra kendimizin de ekleyebileceği faktörler ile satış verilerini kıyaslamamız ve aralarında korelasyon bulunan faktörleri önem sırasına göre belirlememiz beklenmektedir. Faktörlerin satışlar üzerindeki etkileri belirlenerek bir tahmin modeli oluşturmamız beklenmektedir. Bu tahmin modelini belirli bir veri seti üzerinde oluşturup daha sonra elimizdeki test verisine bu modeli uygulayarak doğru veriye yaklaşımımız kontrol edilecektir. Projedeki temel amaç bu test verisine olabildiğince az hata payıyla yaklaşımdır. Projede hata payı ölçütü olarak daha sonra anlatılacak olan RMSPE(Root Mean Square Percentage Error) fonksiyonu kullanılacaktır.

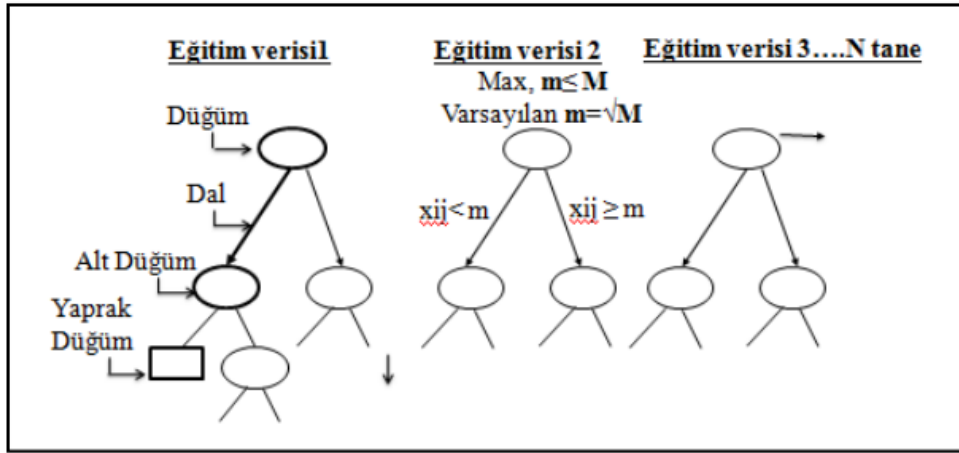
Projedenin uygulanabilmesi için Python dili kullanılmıştır. Python dilinin bizlere sağladığı Skylearn kütüphanesi başta olmak üzere Pandas, Numpy, Matplot, Seaborn kütüphaneleri projede kullanılmıştır. Python dili ve kütüphaneleri sayesinde bu kadar büyük bir veri kolayca işlenip yorumlandırılmıştır. Raporun ilerleyen kısımlarında veri kümesi ile ilgili analizler, kullanılan algoritmalar ve kütüphaneler ile ilgili bilgiler, projenin sonuçları detaylı olarak incelenecektir.

2 LİTERATÜR TARAMASI

2.1 Random Forest Regresyonu

Random Forest regresyonu, hiperparametre ayarı yapmadan bile, iyi sonuçlar verebilen, kolay uygulanabilir bir makine öğrenmesi algoritmasıdır. Basit olması ve iyi sonuç vermesi sebebiyle en çok kullanılan algoritmalarından biridir. Özellikle sınıflandırma ve regresyon işlevleri için kullanılır. Random Forest denetimli bir öğrenme algoritmasıdır. Adından da anlaşılacağı üzere rasgele bir orman oluşturur. Bu orman çoğu zaman "bagging" yöntemiyle eğitilen karar ağaçları topluluğudur. Bagging yöntemi ise öğrenme modellerinin bir kombinasyonunun genel sonucu artırmasıdır. Özetle random forest birden fazla karar ağacını oluşturur ve doğru tahmini yapabilmek için onları birleştirir. Random forest algoritmasını avantajlı kılan taraf onun hem sınıflandırma hem regresyon problemleri için kullanılabilmesi oluşudur. [2]

Random Forest regresyonunda oluşturulan eğitim verileri kullanılarak belirlenen bölünme ölçütlerine göre düğümler dallara ayrılmakta ve ağaç yapıları oluşmaktadır. Şekil 2.1.1' de Random Forest sınıflandırıcısında belirlenen en uygun bölünme pozisyonlarına göre oluşturulan ağaç yapısı örneği gösterilmiştir. [3]



Şekil 2.1.1. Random Forest Ağaç Yapısı[3]

2.2 XGBoost Regresyonu

Random Forest algoritmasında da değindiğimiz bagging bazı modellerin ortalama tekniklerini kullanıp birleştirerek elde ettiğimiz toplama tekniğidir. Bagging' de tahminler bağımsız olarak ele alınır. Bu modelde ele aldığımız boosting ise tahminlerin bagging yönteminin aksine sırayla yapılmaktadır. Bu nedenle bir gözlemin sonraki modellerde de görülme olasılığı eşit değildir. En yüksek hataya sahip olanlar en çok görünenlerdir. Bu yüzden gözlemler bootstrap işlemine göre değil hata oranına göre seçilmektedir. [4] XGBoost algoritması sınıflandırma ve regresyon için oluşturulmuş bir makine öğrenmesi tekniğidir. XGBoost zayıf tahmin modellerinin biraraya gelmesiyle karar ağaçlarının oluşturmuş olduğu bir modeldir. Modellerin amacı denetlenebilir bir kayıp fonksiyonu tanımlamaktır. Modelin hedefi bu kaybı en aza indirmektir. Şekil 2.2.1' de XGBoost için tanımlanan kayıp fonksiyonunu görebilmekteyiz. Model kayıp fonksiyonunu sıfıra yaklaştıracak şekilde tahminlerini güncellemektedir. [4]

XGBoost için kayıp fonksiyonu MSE olarak tanımlanır. Ortalama kare hatası(MSE) Şekil aşağıdaki gibi tanımlanmaktadır.

Kayıp = MSE = $\sum (y_i - y_i^p)^2$
 y_i = i. hedef değeri, y_i^p = i. tahmin değeri, $L(y_i, y_i^p)$ kayıp fonksiyonudur.[4]

Regresyondaki amaç MSE değerini minimuma indirmek, bir başka söylemle sıfıra yaklaştırmaktır. Bu sebeple aşağıdaki işlemler de uygulanmaktadır.

α' nın öğrenme oranı olduğu varsayılırsa, $\sum (y_i - y_i^p)$ hataların toplamıdır.
 $y_i^p = y_i^p - \alpha * 2 * \sum (y_i - y_i^p)$ formülüyle toplam hata hesaplanır.[4]

2.3 NumPy

NumPy, Python programlama dili için oluşturulmuş bir kütüphanedir. Numpy temel olarak matematiksel işlemleri gerçekleştirmeye yarar. Bunun yanısıra çok boyutlu diziler ve matrisler için destek ekler ve bu dizilerde çalışacak geniş bir üst düzey matematiksel işlev koleksiyonu içerir. NumPy'nin atası Numeric, aslen Jim Hugunin tarafından diğer birkaç geliştiricinin katkılarıyla yaratıldı. 2005 yılında Travis Oliphant, rakip Numarray'in özelliklerini geniş kapsamlı değişikliklerle Numeric'e dahil ederek NumPy'yi yarattı. NumPy açık kaynak kodlu bir yazılımdır ve birçok projelere birçok katkıda bulunur. [5]

2.4 Pandas

Pandas adı, aynı paneller için birden fazla zaman diliminde gözlemler içeren veri kümeleri için bir ekonometri terimi olan "panel verileri" teriminden türemiştir. [6] Bilgisayar programlamada Pandas, veri işleme ve analiz amacıyla Python programlama dili için yazılmış bir yazılım kütüphanesidir. Özellikle, sayısal tabloları ve zaman serilerini değiştirmek için veri yapıları ve işlemleri sunar. Üç fıkra BSD lisansı altında yayınlanan ücretsiz bir yazılımdır. [7]

Pandas'ın sağladığı avantajlar: [

- Endekslemeli veri işleme için DataFrame nesnesi oluşturmak
- Bellek içi veri yapıları ve farklı dosya formatları arasında veri okuma ve yazma araçları sağlaması
- Veri hizalama ve eksik verilerin entegre yönetimi
- Veri setlerinin yeniden şekillendirilmesi ve döndürülmesi
- Etiket tabanlı dilimleme, fantezi indeksleme ve büyük veri kümelerinin altkümesine erişebilme
- Veri yapısı sütun ekleme ve silme
- Veri kümesi üzerinde gruplama, birleştirme işlemleri
- Zaman serilerinin işlevsellendirilmesi
- Veri filtrasyonu [8]

2.5 Sklearn

Scikit-learn, Python programlama dili için ücretsiz bir yazılım makinesi öğrenme kütüphanesidir. Destek vektör makineleri, random forest regresyonu , gradyan artırma, k-means ve DBSCAN dahil çeşitli sınıflandırma, regresyon ve kümeleme algoritmalarına sahiptir. Python sayısal ve bilimsel kütüphaneleri NumPy ve SciPy ile birlikte çalışacak şekilde tasarlanmıştır. [9]

Sklearn, scikit-learn'ın kısaltmasıdır. Sklearn, Python için bir makine öğrenme modülüdür. Sklearn, Python'da bir arayüze sahip birçok algoritmadan oluşur. SciPy ve SciPy üzerine kurulu Sklearn, Sklearn'ün kullanımını belirtmek için kurulmalıdır. Sklearn çoğu veri madenciliği görevini içerir. Ayrıca, öğrenmeyi kolaylaştıran güzel bir dokümantasyona sahiptir.

2.6 IPython Notebook

IPython (Interactive Python), başlangıçta Python programlama dili için geliştirilmiş, içe yönlendirme, zengin medya, kabuk sözdizimi, sekme tamamlama ve geçmiş sunan çoklu programlama dillerinde etkileşimli hesaplama için bir komut kabuğudur. IPython aşağıdaki özellikleri sunar:

- Etkileşimli kabuk (terminal ve Qt tabanlı).
- Kod, metin, matematiksel ifadeler, satır içi çizimleri ve diğer ortamları destekleyen tarayıcı tabanlı bir dizüstü bilgisayar arayüzü.
- Etkileşimli veri görselleştirme ve GUI araç setlerinin kullanımı için destek.
- Kendi projelerine yüklemek için esnek, gömülebilir tercümanlar.
- Paralel hesaplama araçları.

2.7 Spyder

Spyder, Python dilinde bilimsel programlama için açık kaynaklı bir çapraz platform entegre geliştirme ortamıdır (IDE). Spyder, NumPy, SciPy, Matplotlib, Pandas, IPython, SymPy ve Cython gibi diğer açık kaynaklı yazılımlar da dahil olmak üzere bilimsel Python yığnında öne çıkan paketlerle bütünleşir. [11] MIT lisansı altında sürümü dağıtılır. [12]

2.8 Matplotlib

Matplotlib, Python programlama dili ve sayısal matematik uzantısı olan NumPy için bir çizim kütüphanesidir. SciPy matplotlib'den yararlanır. [13]

3 ANALİZ VE MODELLEME

3.1 Veri Seti Analizi

Bu bölümde elimizdeki veri seti üzerinde analizlerimizi göstereceğiz. Eğitim veri seti üzerinde bulunan özellikler ve aralıkları Tablo 3.1.1' de gösterildiği gibidir.

Özellik	Aralık
Store ID	1 – 1115
Date	1 Ocak 2013 – 31 Temmuz 2015
Sales	\$0 – \$41,551
DayOfWeek	1 - 7
Customers	0 – 7388
Open	0, 1
School Holidays	0, 1
State Holidays	0, 1, 2, 3
Store Type	a, b, c, d
Product Assortment	a, b, c
Competitor Distance	100 – 76,000 metre
Date Since Competitor Open	1 Ocak 2013 – 31 Temmuz 2015
Promo	0, 1
Existence of Promo2	0, 1
Date Since Promo2 Began	1 Ocak 2013 – 31 Temmuz 2015
Promo Interval	Jan – Dec

Tablo 3.1.1: Eğitim veri setindeki özellikler ve aralıkları

- "Store ID", her mağazanın belirli bir ID numarası olur. Bu mağazanın kimliği niteliğindedir. Satışları değerlendirirken mağazanın ID numarasını ele alıp, ID numarasına göre kıyaslama yapılmaktadır.
- "Date" değişkeni satışın yapıldığı tarihi belirler. Projemizde zaman aralığı 1 Ocak 2013 ile 31 Temmuz 2015 aralığını kapsamaktadır.
- "Sales" değişkeni yapılan günlük satışın toplam miktarını göstermektedir. Bu tutar dolar cinsindedir.
- "DayOfWeek" değişkeni haftanın hangi günü olduğunu belirtmektedir. 1 ile 7 arasında değer alır.
- "Customers" değişkeni o gün içerisinde mağaza satışlarında rol alan müşteri sayısını belirtmektedir.
- "Open" değişkeni mağazanın belirlenen tarihte açık olup olmadığını kontrol etmektedir. Kapalı ise 0, açık ise 1 değerini almaktadır.
- "School Holidays" değişkeni belirlenen tarihte okul tatilinin olup olmadığını kontrol etmektedir. Kapalı ise 0, açık ise 1 değerini almaktadır.
- "State Holidays" değişkeni belirlenen tarihte resmi tatil olup olmadığını kontrol etmektedir. Tatil yoksa 0, resmi tatillerde "a", paskalya tatilinde "b", Noel tatilinde "c" değerini almaktadır.
- "Store Type" değişkeni mağazanın türünü belirlemektedir. "a", "b", "c", "d" türünde 4 farklı mağaza tipi bulunmaktadır.
- "Product Assortment" değişkeni ürün çeşitliliğini belirtmektedir. "a" basit dereceyi, "b" ekstra dereceyi, "c" ise genişletilmiş ürün çeşitliliğini belirtmektedir.

- "Competitor Distance" değişkeni mağazanın en yakın rakip mağazaya olan uzaklığını belirtmektedir.
- "Date Since Competitor Open" değişkeni rakip mağazanın açıldığı tarihi belirtmektedir.
- "Promo" değişkeni belirlenen tarihte belirlenen mağazada promosyon olup olmadığını belirtmektedir. Promosyon yok ise 0, var ise 1 değerini alır.
- "Promo2" değişkeni belirlenen tarihte belirlenen mağazada ikinci promosyon olup olmadığını belirtmektedir. Promosyon yok ise 0, var ise 1 değerini alır.
- "Since Promo2" değişkeni promosyonun başladığı tarihi belirtmektedir.
- "Promo Interval" değişkeni yapılan promosyonun tarih aralığını belirtmektedir.

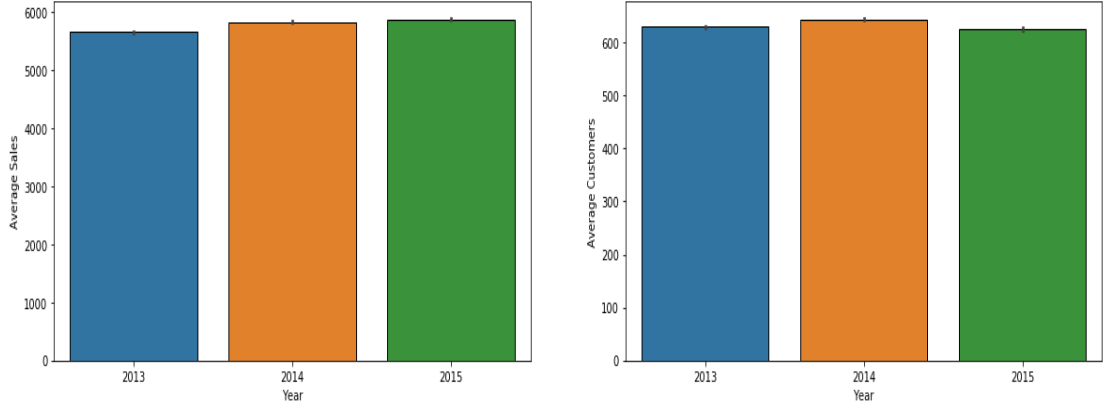
İstatistik	Değerler
Veri setinin büyüklüğü	1017209
Test veri setinin büyüklüğü	41088
Toplam mağaza sayısı	1115
Eğitim verisinin zaman aralığı	2013-01-01 ile 2015-07-31
Test verisinin zaman aralığı	2015-08-01 ile 2015-09-17

Tablo 3.1.2: Veri setinin özellikleri

Tablo 3.1.2' de elimizdeki veri setindeki ögeler için analizler gösterilmektedir. Tabloda da belirtildiği gibi 1115 mağazadaki 1 Ocak 2013 ile 31 Temmuz 2015 arasındaki 1017209 adet satışın değerlerini inceleyip 1 Ağustos 2015 ile 17 Eylül 2015 tarihi arasındaki değerlerine yaklaşıma çalışacağız.

3.2 Veri Analizi

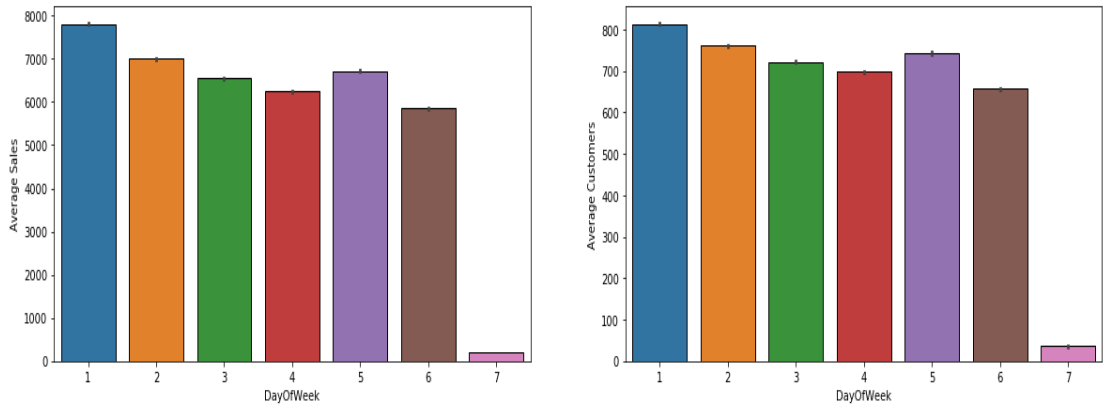
- Ortalama Müşteri Sayısı ve Ortalama Satış Analizi



Resim 3.2.1: Ortalama satış ve ortalama müşteri sayısı

Resim 3.2.1' de görüldüğü üzere yıllara göre satış değerleri birbirinden çok fazla uzaklaşmıyor. 2013' ten 2015' e doğru gelirken ortalama müşteri sayısı azalmış olsa bile ortalama satış miktarında artış görülmektedir.

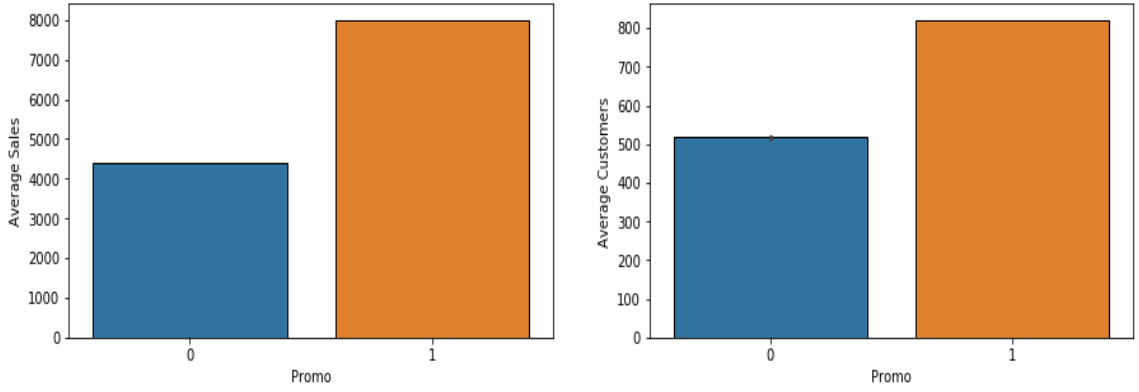
- "DayOfWeek" Analizi:



Resim 3.2.2: Günlere göre satış grafiği

Resim 3.2.2' de görüldüğü gibi satışlar pazartesi günlerinde en yüksek değerine ulaşmıştır. Resimde ayrıca dikkat çeken özellik pazar günleri ortalama müşteri ve ortalama satış miktarının diğer günlere göre çok az olmasıdır. Bu da dükkanların çoğunluğunun pazar günleri kapalı olduğunu göstermektedir.

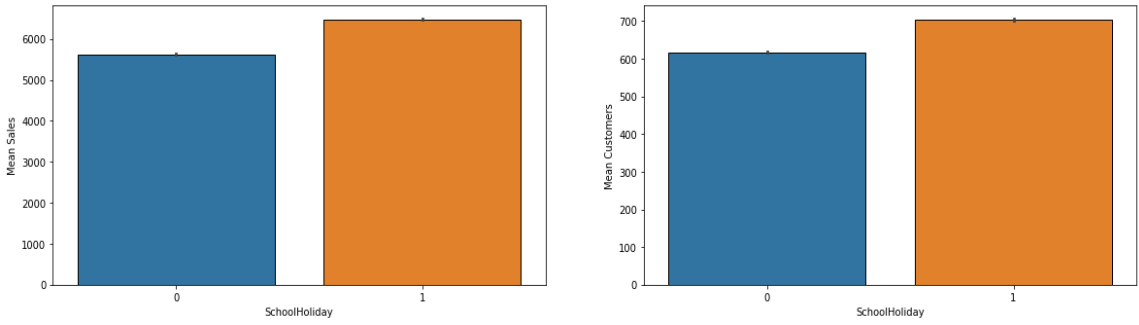
- "Promo" Analizi



Resim 3.2.3: Promosyon değerine göre ortalama satış ve müşteri grafiği

Resim 3.2.3' te görüldüğü üzere promosyon olan günlerdeki ortalama satış ve ortalama müşteri değerinde gözle görülür bir fark vardır. Bu da promosyonla satışların direk alakalı olduğunu göstermektedir.

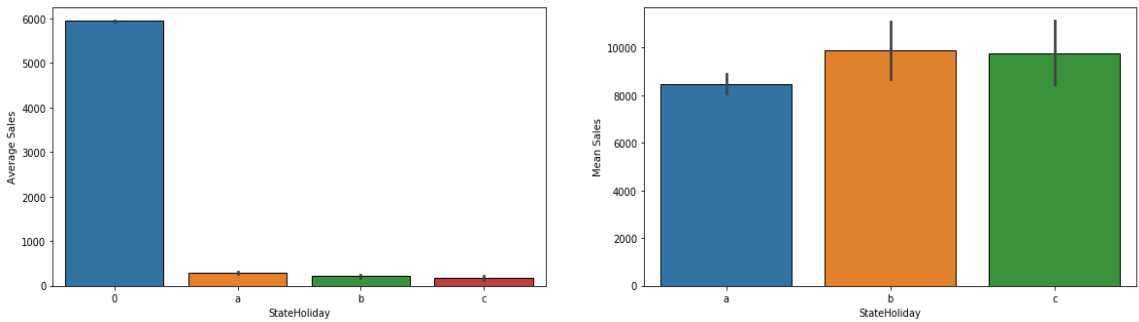
- "SchoolHoliday" Analizi



Resim 3.2.4: Okul tatili değerine göre ortalama satış ve müşteri grafiği

Resim 3.2.4' te görüldüğü üzere okul tatili olması ortalama satış ve ortalama müşteri sayısında artışa sebep olmuştur. Buradan da bir korelasyon elde edilebilmektedir.

- "StateHoliday" Analizi

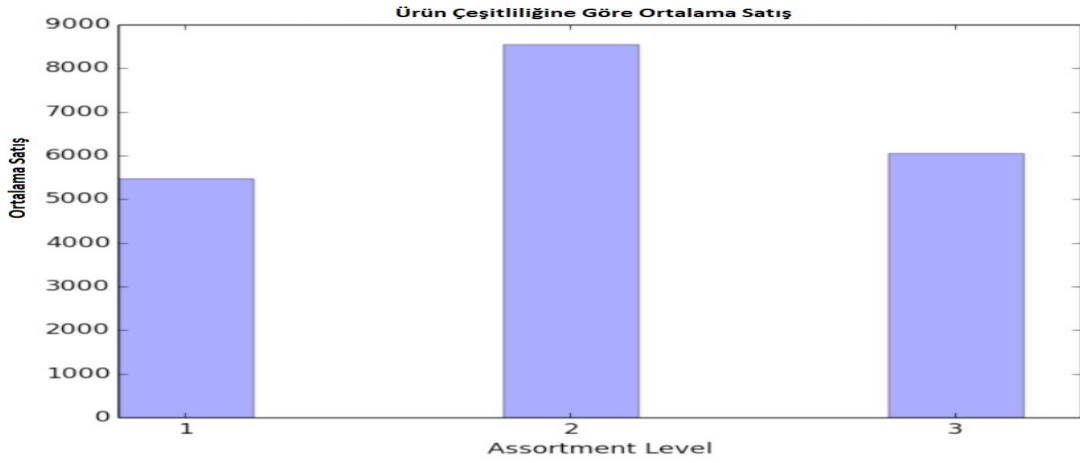


Resim 3.2.5: Özel tatil değerine göre ortalama satış ve müşteri grafiği

Resim 3.2.5' te sol kısımda tatil olmayan günlerdeki ortalama satış değeri ayrıca hesaplanmış olup bu değer 5900 civarında bulunmuştur. Resim 3.2.5' te sağ kısımdaki

grafikte ise a sütunu resmi tatilleri, b sütunu paskalya tatilini, c kısmı Noel tatilini sembol etmektedir.

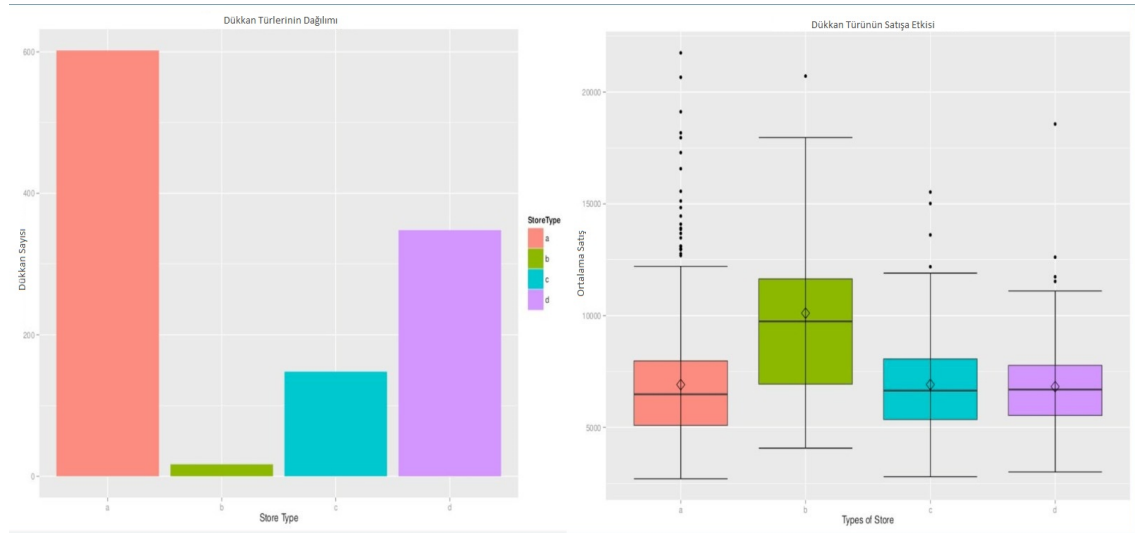
- "Assortment" Analizi



Resim 3.2.6: Ürün çeşitliliğinin ortalama satış üzerinde etkisi

Resim 3.2.6' da mağazadaki ürün çeşitliliğinin satışlar üzerine etkisi incelenmiştir. 1 sütununda normal ürün çeşitliliği, 2 sütununda ekstra ürün çeşitliliği, 3 sütununda ise genişletilmiş ürün çeşitliliği satış sonuçları analiz edilmiştir.

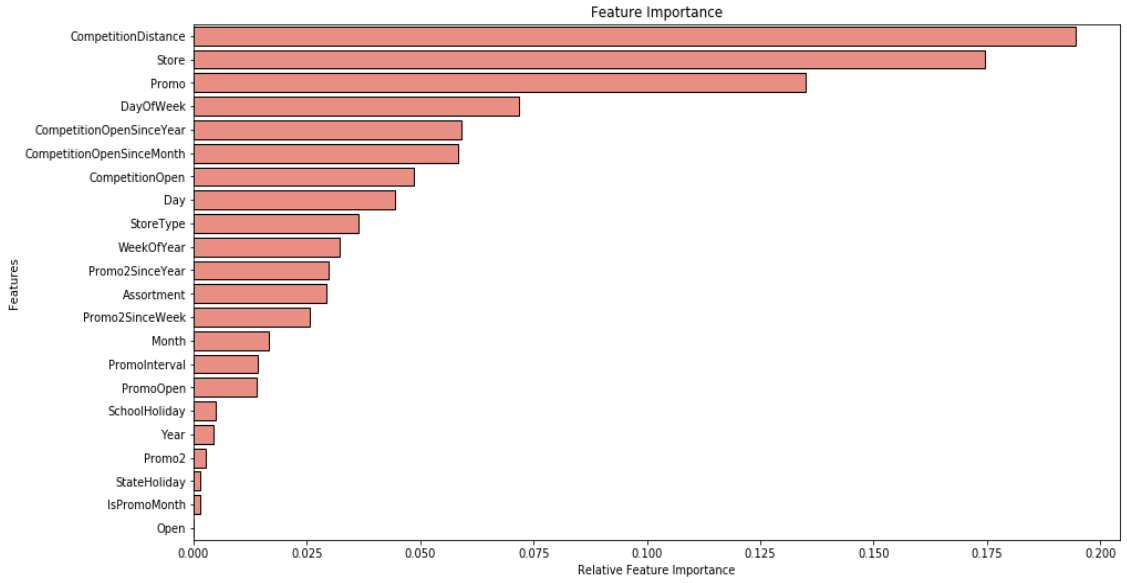
- "Store Type" Analizi



Resim 3.2.7: Mağaza türüne göre mağaza sayısı ve ortalama satış

Resim 3.2.7' de en fazla a türünde mağaza olduğunu en az da b türünde mağaza olduğunu görmekteyiz. Dikkat çekici özellik ise b türünde az mağaza olduğu halde en çok satışın bunlar tarafından yapılmasıdır.

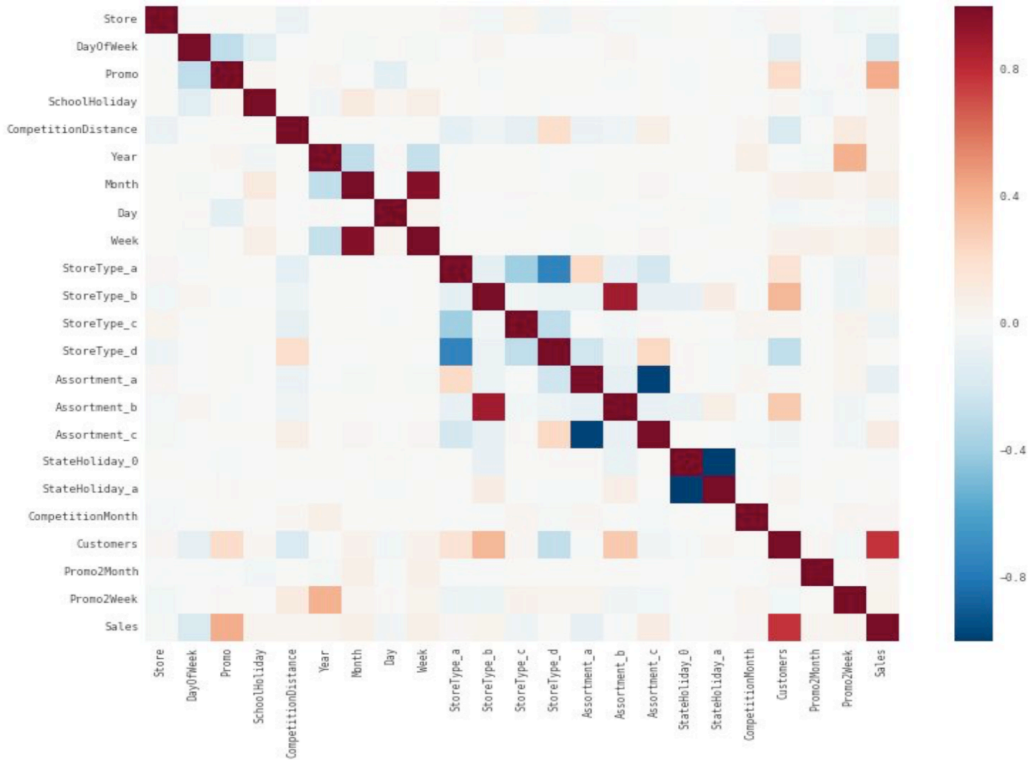
- Özelliklerin Önemi Analizi



Resim 3.2.8: Özelliklerin önem grafiği

Resim 3.2.8' de veri kümemizin özelliklerinin Random Forest regresyonuna göre önem sırasının çıktısını görmekteyiz. Yapılan analize göre Competition Distance, Store ve Promo en önemli etken olarak görülmüştür.

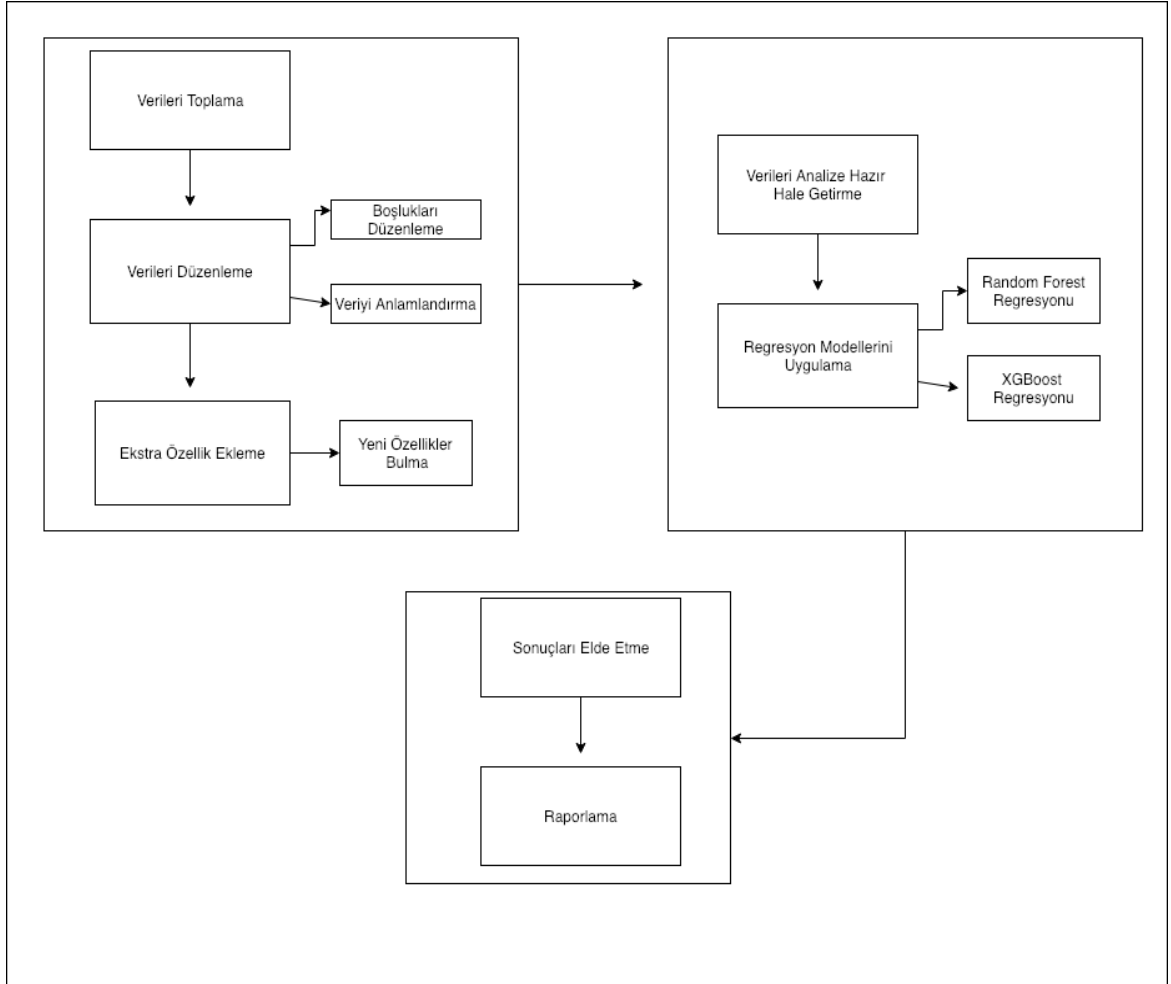
- Korelasyon Tablosu Analizi



Resim 3.2.9: Korelasyon tablosu

Resim 3.2.9' da özelliklerin birbiriyle alakalı olup olmadığı gösterilmektedir. Sıcak renkler aradaki ilişkiyi belirtirken maviye yakın renkler ise ilişkinin azaldığını belirtmektedir.

3.3 Proje Diyagramı



Resim 3.3.1: Projenin akış diyagramı

Projenin akış diyagramı Resim 3.3.1' deki biçimdedir. Temel olarak 3 ana bölümden oluşmaktadır. Bunlar Veri Düzenleme Bölümü, Regresyon Modelini Uygulama bölümü, Sonuçları toplama bölümüdür. Verileri Düzenleme Bölümü' nde veriler toplanır. Verideki boşluklar belirlenir. Bu boşluklar uygun şekilde anlamlandırılır. Boşlukların yanısıra veride matematiksel anlama gelmeyen ifadeler de bulunabilir. Bu ifadeler de modelin anlayacağı şekilde sayısal biçime getirilir. Özellikler yetersizse veriden veya dışarıdan özellikler türetilip veriye entegre edilerek veri modele uygulanmaya hazır hale getirilir. Uygun modellerle veri işleme konularak sonuçlar beklenir. Bizim modelimizde Random Forest Regresyonu ve XGBoost regresyonu kullanılmıştır. En son sonuçlar değerlendirilir. Hata payı hesaplanıp raporlama yapılmaktadır.

4 PROJENİN UYGULANMASI

4.1 Verinin Temizlenmesi

4.1.1 NaN verilerin düzenlenmesi

İlk olarak klasörümüzden eğitim verisi olarak kullanılacak "train.csv", mağaza bilgilerinin olduğu "store.csv" ve test verilerinin olduğu "test.csv" dosyaları okunmuştur. Veriler okunduktan sonra train ve store verilerinde boşluklar tespit edilmiştir. Bu boşluklar değerlendirme yapılırken kullanılmamıştır. Bu boşluklar değerlendirilmede kullanılırsa sonuçları olumsuz etkilerler.

```
138 dataset=dataset2.copy()
139 # REMOVE NANS
140 dataset.fillna(0, inplace=True)
141 dataset.loc[dataset.Open.isnull(), "Open"] = 1
```

Resim 4.1.1.1: Verisetindeki NaN veriler düzenlenmiştir

4.1.2 Açık ve satış yapan mağazaların alınması

İkinci olarak verisetinde Resim 4.1.2.1' de görüldüğü gibi açık olan ve satış yapmış mağazalar inceleme altına alınmıştır. Diğer mağazalar sonucu çok farklı etkileyeceği için analize alınmamıştır.

```
#READ FILE
dataset = pd.read_csv("train.csv", low_memory=False)
store = pd.read_csv("store.csv")
test = pd.read_csv("test.csv")
#SELECT ONLY OPEN STORE AND SALES>0
dataset2 = dataset[dataset["Open"] != 0]
dataset2 = dataset2[dataset2["Sales"] > 0]
dataset2 = pd.merge(dataset2, store, on="Store")
#GET FOR EXTRA FEATURES
```

Resim 4.1.2.1: Verisetindeki açık ve satış yapan mağazalar

4.1.3 Tarih özelliğini parçalama

Veride kullanacağımız tarih özelliği bilgisayarın anlayacağı biçimde eklenmemiştir. Bu değişkeni modelin anlayacağı biçimde ayırıp, o şekilde kullanmamız gerekmektedir. İlk olarak "Date" özelliği datetime nesnesine çevirilmektedir. Daha sonra bu parçalardan yıl, ay, gün, özellikleri elde edilmektedir.

```
#SPLIT DATETIME
```

```
dataset["Date"] = pd.to_datetime(dataset["Date"], errors="coerce")
dataset["Year"] = dataset.Date.dt.year
dataset["Month"] = dataset.Date.dt.month
dataset["Day"] = dataset.Date.dt.day
dataset["DayOfWeek"] = dataset.Date.dt.dayofweek
dataset["WeekOfYear"] = dataset.Date.dt.weekofyear
```

Resim 4.1.3.1: Verisetindeki tarih ögesinin parçalanması

4.1.4 Haritalama işlemleri

Verisetindeki bazı özellikler bilgisayarın anlayacağı matematiksel semboller yerine a,b,c,d gibi harflerle sembolize edilmişlerdir. Bunları modele entegre edebilmek için anlamlı hale getirmemiz gerekmektedir. Bu yüzden uygun sütunlara haritalama işlemini gerçekleştirmemiz gerekir. Resim 4.1.4.1'de gösterildiği üzere haritalama ile StoreType, Assortment, StateHoliday özelliklerinin içerisindeki a yerine 1, b yerine 2, c yerine 3, d yerine 4 sayıları konularak veri daha anlamlı hale getirilmiştir.

```
#SOME FEATURES CONSIST OF A B C D MAPPING THEM
mapp = {'0':0, 'a':1, 'b':2, 'c':3, 'd':4}
dataset.StoreType.replace(mapp, inplace=True)
dataset.Assortment.replace(mapp, inplace=True)
dataset.StateHoliday.replace(mapp, inplace=True)
```

Resim 4.1.4.1: Verisetindeki tarih ögesinin parçalanması

4.1.5 Süre aralığının anlamlandırılması

Verisetindeki bazı sürelerin anlamsız olduğu kanısına varılmıştır. Örneğin 1 yıl 3 aydır açık olan bir rakip mağaza ile 2 yıl 3 aydır açık olan bir mağazanın kıyaslanması küsürat olan 3 aya göre yapılmaktaydı. Bu yüzden "CompetitionOpen", "PromoOpen" özelliklerinin verilerinin daha anlamlı hale getirilmesi gerekmektedir. Bu sebeple bu kıyaslamanın ay üzerinden yapılması mantıklı bulunmuştur. Rakip mağaza 1 yıl 3 aydır açık ise o özellik kısmına 12 ay ve 3 ayın toplamı olan 15 sayısı yazılmaktadır. Böylelikle veriler arasındaki ilişki daha da anlamlı hale getirilerek bir korelasyon olup olmadığı kontrol edilecektir.

```
#CALCULATE OPEN TIME BY THE VALUE OF MONTHS
dataset["CompetitionOpen"] = 12 * (dataset.Year - dataset.CompetitionOpenSinceYear) + (dataset.Month - dataset.CompetitionOpenSinceMonth)
#PROMO OPEN BY THE VALUE OF MONTHS
dataset["PromoOpen"] = 12 * (dataset.Year - dataset.Promo2SinceYear) + (dataset.WeekOfYear - dataset.Promo2SinceWeek) / 4.0
dataset["PromoOpen"] = dataset.PromoOpen.apply(lambda x: x if x > 0 else 0)
```

Resim 4.1.5.1: Verisetindeki süre aralığının anlamlandırılması

4.1.6 Tarihin promosyon aralığında olduğunun kontrolü

"IsPromoMonth" değişkeni belirli bir tarih aralığını bize vermektedir. Bu aralık promosyonun uygulandığı aralıktır. Elimizde satışın yapıldığı tarih bulunmaktadır. Böyle bir özellik türeterek satışın yapıldığı zamanda promosyon olup olmadığı kontrol edilmektedir.

```
for interval in dataset.PromoInterval.unique():
    if interval != '':
        for month in interval.split(','):
            dataset.loc[(dataset.PromoInterval == interval), "IsPromoMonth"] = 1
```

Resim 4.1.6.1: Verisetindeki süre aralığının anlamlandırılması

4.2 Ekstra Özellik Türetme

Projede veri kümesinde mevcut sayıda birçok özellik bulunmaktadır. Yeni özellikler eğer incelediğimiz değer için olumlu katkı yapıyorsa projeye eklenmesi proje sonucunda bizi doğru değere daha çok yaklaştırır.

	A	B	C	D	E	F	G	
1	Store,"DayOfWeek","Date","Sales","Customers","Open","Promo","StateHoliday","SchoolHoliday"							
2	1,5,2015-07-31,5263,555,1,1,"0","1"							
3	2,5,2015-07-31,6064,625,1,1,"0","1"							
4	3,5,2015-07-31,8314,821,1,1,"0","1"							
5	4,5,2015-07-31,13995,1498,1,1,"0","1"							
6	5,5,2015-07-31,4822,559,1,1,"0","1"							
7	6,5,2015-07-31,5651,589,1,1,"0","1"							
8	7,5,2015-07-31,15344,1414,1,1,"0","1"							
9	8,5,2015-07-31,8492,833,1,1,"0","1"							
10	9,5,2015-07-31,8565,687,1,1,"0","1"							
11	10,5,2015-07-31,7185,681,1,1,"0","1"							
12	11,5,2015-07-31,10457,1236,1,1,"0","1"							
13	12,5,2015-07-31,8959,962,1,1,"0","1"							
14	13,5,2015-07-31,8821,568,1,1,"0","0"							
15	14,5,2015-07-31,6544,710,1,1,"0","1"							
16	15,5,2015-07-31,9191,766,1,1,"0","1"							
17	16,5,2015-07-31,10231,979,1,1,"0","1"							
18	17,5,2015-07-31,8430,946,1,1,"0","1"							
19	18,5,2015-07-31,10071,936,1,1,"0","1"							
20	19,5,2015-07-31,8234,718,1,1,"0","1"							
21	20,5,2015-07-31,9593,974,1,1,"0","0"							
22	21,5,2015-07-31,9515,682,1,1,"0","1"							
23	22,5,2015-07-31,6566,633,1,1,"0","0"							
24	23,5,2015-07-31,7273,560,1,1,"0","1"							
25	24,5,2015-07-31,14190,1082,1,1,"0","1"							

Resim 4.2.1: Eğitim verisinin görünümü

Resim 4.2.1 de eğitim verisinde veriler "Store" değerine göre sıralanmıştır. Store değerinden sonra tarihsel olarak bir sıralama mevcuttur.

	A	B	C	D	E	F	G	H	I
1	Store,"StoreType","Assortment","CompetitionDistance","CompetitionOpenSinceMonth","CompetitionOpenSinceYear"								
2	1,"c","a",1270,9,2008,0,,,""								
3	2,"a","a",570,11,2007,1,13,2010,"Jan,Apr,Jul,Oct"								
4	3,"a","a",14130,12,2006,1,14,2011,"Jan,Apr,Jul,Oct"								
5	4,"c","c",620,9,2009,0,,,""								
6	5,"a","a",29910,4,2015,0,,,""								
7	6,"a","a",310,12,2013,0,,,""								
8	7,"a","c",24000,4,2013,0,,,""								
9	8,"a","a",7520,10,2014,0,,,""								
10	9,"a","c",2030,8,2000,0,,,""								
11	10,"a","a",3160,9,2009,0,,,""								
12	11,"a","c",960,11,2011,1,1,2012,"Jan,Apr,Jul,Oct"								
13	12,"a","c",1070,,1,13,2010,"Jan,Apr,Jul,Oct"								
14	13,"d","a",310,,1,45,2009,"Feb,May,Aug,Nov"								
15	14,"a","a",1300,3,2014,1,40,2011,"Jan,Apr,Jul,Oct"								
16	15,"d","c",4110,3,2010,1,14,2011,"Jan,Apr,Jul,Oct"								
17	16,"a","c",3270,,0,,,""								
18	17,"a","a",50,12,2005,1,26,2010,"Jan,Apr,Jul,Oct"								
19	18,"d","c",13840,6,2010,1,14,2012,"Jan,Apr,Jul,Oct"								
20	19,"a","c",3240,,1,22,2011,"Mar,Jun,Sept,Dec"								
21	20,"d","a",2340,5,2009,1,40,2014,"Jan,Apr,Jul,Oct"								
22	21,"c","c",550,10,1999,1,45,2009,"Jan,Apr,Jul,Oct"								
23	22,"a","a",1040,,1,22,2012,"Jan,Apr,Jul,Oct"								
24	23,"d","a",4060,8,2005,0,,,""								
25	24,"a","c",4590,3,2000,1,40,2011,"Jan,Apr,Jul,Oct"								

Resim 4.2.1: Mağaza verisinin görünümü

Üretmek istediğimiz özellikler her mağaza için haftalık satış ortalaması, aylık satış ortalaması ve yıllık satış ortalaması verileridir. Bu sebeple eğitim veri kümesi ile mağaza veri kümesini katma(join) yaparak "Store" özelliği ile gruplandırıldı. Böylelikle aynı "Store" değerine sahip mağazalar gruplandırılmış oldu. Gruplanan bu veri kümesi tarihe göre sıralı hale getirildi. Böylelikle ortalamaları hesaplamak daha kolay oldu. Her mağaza için "DayOfWeek" verisi kullanılarak o hafta yapılan satışların ortalaması alınarak o satışların olduğu günlerin satırına "WeekAvg" özelliği olarak eklendi. Daha önce parçaladığımız "Month" ve "Year" verileri kullanılarak "MonthAvg" olarak aylık ortalama, "YearAvg" olarak yıllık ortalama hesaplanmış oldu. Daha sonra bunlar veri kümesine entegre edilmiş oldu. Çıkarım yapılan bu özellikler "extraFeatures.csv" olarak kaydedildi.

	WeekAvg,MonthAvg,YearAvg		
1	5235,4491,4527		
2	5235,4491,4527		
3	5235,4491,4527		
4	5235,4491,4527		
5	5235,4491,4527		
6	5235,4491,4527		
7	3876,4491,4527		
8	3876,4491,4527		
9	3876,4491,4527		
10	3876,4491,4527		
11	3876,4491,4527		

Resim 4.2.2: Üretilen özelliklerin görünümü

Üretilen bu extraFeatures.csv dosyası Resim 4.2.3' te görüldüğü gibi veri kümesine entegre edildi.

```
#LIST FOR EXTRA FEATURES
weekList=[]
monthList=[]
yearList=[]
c=0
#READ EXTRA FEATURES
with open('extraFeatures.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    for row in csv_reader:
        c+=1
        if c==1:
            continue
        weekList.append(int(row[0]))
        monthList.append(int(row[1]))
        yearList.append(int(row[2]))
#ADD TO DATASET
dataset2["WeekAvg"]=weekList
dataset2["MonthAvg"]=monthList
dataset2["YearAvg"]=yearList
```

Resim 4.2.3: Üretilen özelliklerin veri kümesine entegrasyonu

4.3 Regresyonların Uygulanması

4.3.1 Random Forest Regresyonu

Veri kümesi son halini aldıktan sonra daha hızlı işleme sokabilmek için veriyi parçalara bölmek gerekti. Bu nedenle "train_test_split" fonksiyonu ile veri daha küçük parçalara bölünmüş oldu.

```
#DEEPCOPY FUNCTION
def Make_X_y(data, col="y"):
    y = data[col].as_matrix()
    X = deepcopy(data)
    del X[col]
    X = X.as_matrix()
    X, X_v, y, y_v = train_test_split(X, y, test_size=0.2, random_state=8)
    return X, y, X_v, y_v
#SPLIT DATA INTO SETS -> TEST AND VALIDATION
X, y, X_v, y_v = Make_X_y(dataset2, col="Sales")
```

Resim 4.3.1.1: Veri kümesinin parçalara bölünmesi

Daha sonra Resim 4.3.2' de görüldüğü gibi veri regresyon işlemine konuldu. Önemli olan özellikleri bulmak için de "feature_importances_" fonksiyonundan yararlanıldı.

```
#REGRESSION
rf_reg = RandomForestRegressor(n_jobs=-1, random_state=8)
rf_reg.fit(X, y)
#FEATURES
features = rf_reg.feature_importances_
feature_importance = DataFrame(list(reversed(sorted(zip(features, features_list)))))
#CREATE THE IMPORTANTANCE TABLE
```

Resim 4.3.1.2: Regresyon işlemi

4.3.2 XGBoost Regresyonu

Veri temizlendikten sonra Random Forest regresyonunun yanısıra bir de XGBoost regresyonuyla işleme konuldu. XGBoost regresyonunda Random Forest regresyonundan farklı olarak parametrelerin belirlenmesi gerekmektedir. Bu parametreler daha önce yapılan benzer çalışmalardan alınmıştır.

```
#parameters
params = {"objective": "reg:linear",
          "booster" : "gbtree",
          "eta": 0.3,
          "max_depth": 10,
          "subsample": 0.9,
          "colsample_bytree": 0.7,
          "silent": 1,
          "seed": 1301,
          "learning_rate":0.2
        }
num_boost_round = 200
```

Resim 4.3.2.1: XGBoost algoritmasının parametreleri

Random Forest Algoritmasının aksine XGBoost algoritmasında DMatrix kullanılmıştır. Eğitim veri kümesi NumPy kütüphanesinin "log1p" fonksiyonu kullanılarak logaritmik düzeyde temsil edilip, korelasyon böyle aranmıştır.

```
trainX, validX = train_test_split(trainSet, test_size=0.012, random_state=10)
train_y = np.log1p(trainX.Sales)
valid_y = np.log1p(validX.Sales)
D_train = xgb.DMatrix(trainX[features], train_y)
D_valid = xgb.DMatrix(validX[features], valid_y)
```

Resim 4.3.2.2: DMatrix kullanımı

5 SONUÇ VE DEĞERLENDİRME

Random Forest ve XGBoost regresyonlarına konulan veri kümelerimizin test verileriyle karşılaştırılıp bir hata değeri bulunması beklenir. Projedeki amaç bu hata değerini sıfıra yaklaştırmaktır.

Bu projede hata değeri ölçütü olarak RMSPE(Root Mean Square Percentage Error) kullanılmıştır. Projede kullanılan RMSPE fonksiyonu Resim 5.1 de gösterilmiştir.

$$\text{RMSPE} = \sqrt{\frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{y}_i}{y_i} \right)^2}$$

Resim 5.1: RMSPE fonksiyonu[1]

```
#RMSPE FUNCTION
def RMSPE(y,yhat):
    assert len(y)==len(yhat)

    yhat[yhat<=0]=0

    summ=0
    c=0
    for pair in zip(y,yhat):

        fraction=(y-yhat)/y

        fraction_squared=fraction**2
        summ+=fraction_squared[c]
        c+=1
    error=np.sqrt((1/len(yhat))*summ)

    return error
hata=RMSPE(y_v,y_pred)
print("RMSPE Error: %f"%hata)
print(hata)
```

Resim 5.2: RMSPE fonksiyonunun implementasyonu

5.1 Random Forest Regresyonunun Sonucu

Eğitim verisi okunup düzenlendikten sonra regresyona konularak bir tahmin kümesi elde edilmiştir. Bu tahmin kümesi XGBoost' un aksine reel değerlerden oluşmaktadır. Bu tahmin kümesi daha sonra RMSPE fonksiyonuyla işleme konulup test kümesiyle arasındaki hata payı ölçülmüştür.

0	
0	3654.3
1	10913.9
2	6104
3	5111.9
4	5221.9
5	13527.5
6	7803
7	4514.8
8	9721.6
9	7747.9

Resim 5.1.1: Tahmin kümesinin görünümü

Elimizdeki bu tahmin kümesini, test verileriyle kıyaslanmasından ortaya çıkan RMSPE değeri 0.146836 olarak bulunmuştur.

5.2 XGBoost Regresyonunun Sonucu

XGBoost regresyonuna koyduğumuz eğitim verilerinden bir tahmin kümesi elde edilmiştir. Bu küme eksponansiyel değerler biçiminde üretilmiştir. RMSPE değeri belirlenirken bu kümedeki her indeksin eksponansiyel olarak değerini veren "expm1" fonksiyonu kullanılmıştır. Şekil 5.2.1' de bu değerlerin logaritmik hali görülebilmektedir.

```
yhat = gbm.predict(xgb.DMatrix(validX[features]))
error = rmspe(validX.Sales.values, np.expm1(yhat))
print('RMSPE: {:.6f}'.format(error))
```

Resim 5.2.1: RMSPE fonksiyonu

0	
0	8.2072
1	8.46713
2	8.56448
3	9.04917
4	8.52179
5	8.9245
6	8.1747
7	8.94494
8	9.07072
9	8.57303

Resim 5.2.2: XGBoost için tahmin kümesinin görünümü

Elimizdeki bu tahmin kümesinin e^8 değeri olarak değerlendirilmesi gerektiğini ve 0. değerin $e^{8.2072}$ yani 3667.25, 1. değerin 4755.84 olarak hata payının ölçüldüğü bilinmelidir.

```
[166] train-rmse:0.310386 eval-rmse:0.309758 train-rmspe:0.26557 eval-rmspe:0.261566
[167] train-rmse:0.305172 eval-rmse:0.304559 train-rmspe:0.261898 eval-rmspe:0.257828
[168] train-rmse:0.300055 eval-rmse:0.299466 train-rmspe:0.258301 eval-rmspe:0.25416
[169] train-rmse:0.295045 eval-rmse:0.294472 train-rmspe:0.254754 eval-rmspe:0.250548
[170] train-rmse:0.290155 eval-rmse:0.289597 train-rmspe:0.251269 eval-rmspe:0.247014
[171] train-rmse:0.285354 eval-rmse:0.284818 train-rmspe:0.247842 eval-rmspe:0.243523
[172] train-rmse:0.280676 eval-rmse:0.280164 train-rmspe:0.24451 eval-rmspe:0.240131
[173] train-rmse:0.276099 eval-rmse:0.275608 train-rmspe:0.241242 eval-rmspe:0.236798
[174] train-rmse:0.271619 eval-rmse:0.271147 train-rmspe:0.237965 eval-rmspe:0.233506
[175] train-rmse:0.267254 eval-rmse:0.266813 train-rmspe:0.234792 eval-rmspe:0.230315
[176] train-rmse:0.262986 eval-rmse:0.262556 train-rmspe:0.231716 eval-rmspe:0.227167
[177] train-rmse:0.258826 eval-rmse:0.258424 train-rmspe:0.228722 eval-rmspe:0.224106
[178] train-rmse:0.254797 eval-rmse:0.254416 train-rmspe:0.225813 eval-rmspe:0.221123
[179] train-rmse:0.250838 eval-rmse:0.250487 train-rmspe:0.222946 eval-rmspe:0.218206
[180] train-rmse:0.247 eval-rmse:0.246679 train-rmspe:0.220181 eval-rmspe:0.215376
[181] train-rmse:0.243238 eval-rmse:0.242939 train-rmspe:0.21746 eval-rmspe:0.212589
[182] train-rmse:0.239598 eval-rmse:0.23932 train-rmspe:0.214826 eval-rmspe:0.209877
[183] train-rmse:0.236032 eval-rmse:0.235777 train-rmspe:0.212193 eval-rmspe:0.20723
[184] train-rmse:0.232524 eval-rmse:0.232294 train-rmspe:0.20951 eval-rmspe:0.204623
[185] train-rmse:0.229098 eval-rmse:0.228894 train-rmspe:0.207018 eval-rmspe:0.202072
[186] train-rmse:0.225784 eval-rmse:0.225604 train-rmspe:0.20462 eval-rmspe:0.19961
[187] train-rmse:0.222536 eval-rmse:0.222381 train-rmspe:0.202244 eval-rmspe:0.197164
[188] train-rmse:0.219366 eval-rmse:0.219239 train-rmspe:0.199946 eval-rmspe:0.1948
[189] train-rmse:0.216272 eval-rmse:0.216185 train-rmspe:0.197687 eval-rmspe:0.192494
[190] train-rmse:0.21326 eval-rmse:0.213204 train-rmspe:0.195444 eval-rmspe:0.190252
[191] train-rmse:0.210308 eval-rmse:0.210279 train-rmspe:0.19328 eval-rmspe:0.188037
[192] train-rmse:0.207466 eval-rmse:0.207472 train-rmspe:0.191162 eval-rmspe:0.185919
[193] train-rmse:0.204681 eval-rmse:0.204715 train-rmspe:0.189148 eval-rmspe:0.183839
[194] train-rmse:0.201948 eval-rmse:0.20201 train-rmspe:0.187151 eval-rmspe:0.181794
[195] train-rmse:0.199292 eval-rmse:0.199385 train-rmspe:0.185156 eval-rmspe:0.179821
[196] train-rmse:0.196695 eval-rmse:0.196815 train-rmspe:0.183275 eval-rmspe:0.177885
[197] train-rmse:0.194188 eval-rmse:0.194342 train-rmspe:0.181382 eval-rmspe:0.176025
[198] train-rmse:0.191734 eval-rmse:0.191911 train-rmspe:0.179621 eval-rmspe:0.174208
[199] train-rmse:0.189338 eval-rmse:0.189538 train-rmspe:0.177864 eval-rmspe:0.172421
Validating...
RMSPE: 0.172421
```

Resim 5.2.2: XGBoost için sonuç görünümü

Resim 5.2.2' de XGBoost regresyonunun son aşamaları gözükmektedir. 200 iterasyon sonucu RMSPE değeri olarak 0.172421 hata payıyla sonuca yaklaşmış olduk.

5.3 Sonucun Görselleştirilmesi

Resim 5.3.1' de veri kümemizdeki satış değerlerinin aylık olarak değerlerini görebilmekteyiz. Alttaki grafikte ise satış değerlerinin yüzdesel olarak değişikliğini görebilmekteyiz. Grafikteki en önemli ayrıntı Aralık 2013 ve Ocak 2015' te önceki aylarda satışlarda ciddi miktarda düşüşlerin yaşandığıdır.



Resim 3.2.10: Ortalama satış ve satışın yüzdelik değişiminin grafiği

Sonucu görselleştirmek adına Şekil 5.3.2' de görüldüğü üzere rastgele 3 adet mağaza numarası seçip bu mağazalar için Train.Sales değeriyle Prediction değerleri grafiğe dökülmüştür.

```
train = pd.DataFrame(data = y_train)
train['Prediction']=yhat
train = pd.merge(X_train,train, left_index= True, right_index=True)
train['Ratio'] = train.Prediction/train.Sales
train['Error'] =abs(train.Ratio-1)
train['Weight'] = train.Sales/train.Prediction
train.head()
col_1 = ['Sales','Prediction']
col_2 = ['Ratio']
L=np.random.randint( low=1,high = 1115, size = 3 )
print(format(train.Ratio.mean()))
for i in L:
    s1 = pd.DataFrame(train[train['Store']==i],columns = col_1)
    s2 = pd.DataFrame(train[train['Store']==i],columns = col_2)
    s1.plot(title = 'Gerçek ve satış değerlerinin karşılaştırılması: Mağaza No.store {}'.format(i),figsize=(12,4))
train.sort_values(['Error'],ascending=False,inplace= True)
```

Resim 5.3.1: Veri görselleştirmesi için gereken kod










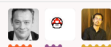
Rastgele mağaza numaralarımız, 124, 321 ve 454' tür. Bu mağazalar için satış ve tahmin değerleri aşağıdaki gibi gösterilmiştir. Buradan verilerin birbirine ne kadar yakınsandığı görülebilmektedir.



Resim 5.3.2: Satış ve tahmin verileri görselleştirmesi

6 SONUÇ

Rossmann Store Sales adlı Kaggle yarışmasını ele alan projede train, store ve test verilerini okunup boşluklar temizlendi. Daha sonra haritalama ve özellik taraması yapıldı. Ekstra özellikler eklendi. Daha sonra regresyona hazır olan veri kümesi regresyona konulup bir tahmin verisi elde edildi. Bu tahmin verisi ile test verisi karşılaştırılarak bir hata fonksiyonu olan RMSPE değeri bulundu. Resim 6.1' de Kaggle yarışmasının ilk 10 sıralaması ve RMSPE değerleri görülmektedir.

1	▲1	Gert		0.10021	19	3y
2	▲1	NimaShahbazi		0.10386	196	3y
3	▲10	Neokami Inc		0.10583	40	3y
4	▲16	Russ W		0.10621	126	3y
5	▲10	MIPT + PZAD		0.10763	195	3y
6	▲96	João N. Laia		0.10771	14	3y
7	▼6	SDNT		0.10784	289	3y
8	▲47	Evdilos Ikaria		0.10817	239	3y
9	▲42	Too busy to compete		0.10826	200	3y
10	▲12	NaiveLearners		0.10839	367	3y

Resim 6.1: Kaggle yarışması lider sıralaması[14]

Yapılan regresyonların sonucunda Random Forest regresyonundan RMSPE olarak 0.146836 değeri, XGBoost regresyonundan 0.172421 hata payı alınmış oldu. Kaggle sıralamasında 3303 kişi bulunmaktadır. Bu sonuçlardan en iyisi 0.10021 olup en kötü mantıklı skor tamamen alakasız olan 2.44 değeridir. Mantıklı sonuçlar ön aralıklarda yığılmış bulunmaktadır. Yarışma 3 yıl önce yani 2015 yılında yapılmıştır. Eğer bu sonuçlarla o zaman yarışmaya katılmış olabilseydik Random Forest sonucumuz olan 0.146836 ile 2174. sırada yer alır, 0.172421 hata payı olan XGBoost sonucumuz ile 2549 uncu sırada yer alırdık.

Alınan sonuçlar pek mükemmel olmasa da veriden öğrenme metotları ve makine öğrenimi hakkında projenin katkısı büyüktür.

Sonucu etkileyen faktörler ele alırsa başta verinin temizliği gelir. Böyle bir projeyle ilk defa ilgilenildiği için projede eksiklikler olabilir. Veriye daha profesyonel dokunuşlar yapıp veri daha da anlamlı hale getirilebilirdi. Ayrıca veriye dışarıdan daha fazla özellik eklenerek sonuca daha çok yaklaşıp hata skoru azaltılabilirdi. Kullandığımız regresyon veri modellerinin yanısıra birçok regresyon modeli bulunmaktadır. Değişik veritiplerinde değişik regresyon algoritmalarının daha iyi çözümleme yapabileceği aşıkardır. Bu sebeple bir veri kümesi için aynı regresyon modelini kullanmak yerine farklı regresyon modellerini denemek sonuca katkı sağlayabilir.

7 KAYNAKÇA

- [1] *Rossmann Store Sales*. (2015). Kaggle: <https://www.kaggle.com/c/rossmann-store-sales> adresinden alındı
- [2] Donges, N. *The Random Forest Algorithm* . Towards Data Science: <https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd> adresinden alındı
- [3] Ok, Akar, Gungor (Haziran, 2011). Rastgele Orman Sınıflandırma Yöntemi Yardımıyla Tarım Alanlarındaki ürün Çeşitliliğinin Sınıflandırılması. *TUFUAB 2011 VI. Teknik Sempozyumu*. Antalya.
- [4] Grover, P. (2017, Aralık). *Gradient Boosting from scratch* . Medium: <https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d> adresinden alındı
- [5] *Numpy*. Wikipedia: <https://en.wikipedia.org/wiki/NumPy> adresinden alındı
- [6] McKinney, W. (tarih yok). *pandas: a Foundational Python Library for Data Analysis and Statistics.*, (s. 2).
- [7] *Pandas*. PyData: <https://pandas.pydata.org/pandas-docs/stable/overview.html#license> adresinden alındı
- [8] *Pandas*. Wikipedia: [https://en.wikipedia.org/wiki/Pandas_\(software\)](https://en.wikipedia.org/wiki/Pandas_(software)) adresinden alındı
- [9] *Scikit-Learn*. Wikipedia: <https://en.wikipedia.org/wiki/Scikit-learn> adresinden alındı
- [10] *IPython* Wikipedia: <https://en.wikipedia.org/wiki/IPython> adresinden alındı
- [11] Spyder Review. (2014, Şubat 9). *Techworld*.
- [12] <https://github.com/spyder-ide/spyder/blob/master/LICENSE> adresinden alındı
- [13] Matplotlib: https://matplotlib.org/faq/usage_faq.html#coding-styles adresinden alındı
- [14] <https://www.kaggle.com/c/rossmann-store-sales/leaderboard> adresinden alındı