Müşteriye Özel Fiyat Tahmin Çalışması

Berkay AKAR1

*1Düzce Üniversitesi Bilgisayar Mühendisliği*

**Özet**

Teknolojinin gelişmesi, yarı iletken maddelerin üretiminin ucuzlaması ve bunlarında beraberinde performanslarının ve kullanışlılığının artması ile hayatımızda getirdiği vazgeçilmez bir yer almaya başlayan bilgisayar teknolojilerinin beraberinde getirmiş olduğu yapay zeka teknolojilerinin günlük hayat üzerinden uygulamalarının incelenmesi ve analizlerinin yapılarak sektörel bazlı işleyişinin incelenmesi amaçlanmıştır. Beraberinde yapılan bu çalışmanın amacı yapay zeka teknolojilerinin bir pazar sürecinde müşterilere özgü indirim tahmininde bulunulması amaçlanmış, bu amaçlar doğrultusunda farklı platformlarda geliştirilen bu çalışmadan çıkan algoritmalar yorumlanarak sektörel açıdan var olan uygulamalarla karşılaştırılması hedeflenmiştir ve sonuçları tartışılmıştır.

**Anahtar Kelimeler:** İndirim, Tahminleme, Knime, Sklearn, Veri Bilimi, Müşteri, Davranış

**Abstract**

Computer technologies, which have become an indispensable part of our lives owing to technological advancements, cheaper semiconductor material production, and increased performance and utility, have brought artificial intelligence technologies alongside them. And here, it is intended to evaluate their applications in daily life as well as their operations based on sectors by examining them. At the same time, the goal of this study was to use artificial intelligence to predict discounts specific to customers in a market process, and the algorithms produced on different platforms and obtained from this study were evaluated and compared with current applications in the sectors; the results were discussed.

**Anahtar Kelimeler:** Discount, Prediction, Knime, Sklearn, DataScience, Costumer, Behaivoral

# **1.Giriş:**

Teknolojinin gün geçtikçe gelişmesi, mikroişlemci ve mikro denetleyici modüllerin üretim ücretlerinin azalması, ham maddelerin ucuzlaması ve bu teknolojilerin kullanışlılığının artması, insanların yapabilecekleri iş hızını aşmaları sebebiyle bilgisayar ve mikro denetleyici aygıtların hayatımızda vazgeçilmez bir bütününü oluşturmaktadır. Bu durum bilgisayar bilimlerinin gelişmesine ve başka bilimler ile birleştirilerek insanların hayattaki problemlerine çözüm odağı olması için hızlı bir gelişme ve çözüm süreci getirmiş bulunmaktadır. Bu gelişmeler hayatımıza farklı sorunlara farklı çözümler sunulması ile çok büyük bir talep ve pazar payı oluşturmaktadır.

İşleyiş açısından hayatımızı ne kadar kolaylaştırsa ve kullanışlılığı ile birçok sektörde kullanılsa bile sektörel manada ve kişisel kullanım düşünüldüğünde artan veri sayısı göz önünde bulundurulduğunda her geçen zaman bir önceki zaman diliminden daha fazla bilgi üretilmekte ve tüketilmektedir. International Data Corporation (IDC) bir rapora göre 2011'de genel olarak oluşturulmuş ve dünyadaki kopyalanan veri hacmi yaklaşık olarak 1.8 Zettabyte olarak hesaplanmıştır [3]. IDC’nin yine bir açıklamasına göre tahmini olarak 2025’te bu sayı 175 zettabytes’a ulaşabileceğinin tahminindedir [4]. IDC’nin düzenli olarak paylaştığı senelik üretilen veri grafiklerine bakılacak olursa bu açıklanan değerler baz alınarak her geçen sene bir önceki seneye göre daha fazla bir artış gerçekleşmektedir.

Bu üretim ve tüketimin en başlıca yaygınlaşma sebepleri olarak teknolojilerin yaygınlaşması, kişisel istek veya amaç doğrultusunda yeni istemlerin oluşması, sağlık, askeri, eğitim kurumları çerçevesinde eski bilgilerden yararlanılarak tahminleme ve öngörü istekleri, bireysel kullanıcılara, sosyal medya işleyişine, kişisel tüketim elektroniği kullanımları, gerek kişisel tüketim elektroniği gerekse eski zamanlarda manuel çalışan cihazların akılcılaştırılması ve bu süreç içerisinde işlemlerini yerine getirebilmeleri için sensorlar vasıtası ile kişilerin kendilerinin bile veriler üretebilmelerine[1][2], üretilen verilerden uygulamalar vasıtası ile tahminleme ve tavsiyelerde bulunma başlıca örnekleri olmak üzere sürekli bir veri üretimi ve tüketimin mevcut olduğunun ve verinin her alanda bulunduğunun başlıca göstergesidir.

Düzenli olarak yükseliş gösteren ve geleceğin teknolojisi olan veri bilimi sektörü haliyle birçok alanda çalışmalar sürdürülebilmektedir ve aynı zamanda gerekli veriler işlenerek birçok alanda insan işleyişinin kolaylaştırılması sağlanabilmiştir. En bilindiklerine örnekler vermek gerekirse borsa tahmini, sağlık incelemesi, erken hastalık tespiti, müşteri kredi skorlaması, CRM, müşteri analizi, ürün tavsiyesi başta olmak üzere birçok alanda kullanılmaktadır. Bu makalede amacımız ve çalışmamız veri bilimi ve istatistiksel modellerin kullanılmasının değerlendirilmesinin satış, pazar tahminlemesi, müşteriye özel fiyatlandırma olacağı için ağırlıklı olarak bu konuların üzerinde durulacaktır.

# **2. Müşteri Davranışlarının Analizi Ve Satış İlişkisi:**

Bu makalemizinde ana konusu olan ve üzerinde incelemelerimizi yapacağımız sorun üretilen bu verilerden gerekli istatiksel modeller uygulanarak satış işlemlerinin müşteri ve ürün arasındaki ilişkisinin incelenmesi ve bu ilişki sonucu müşterilerimize ürünleri ne kadar indirim ile satılabileceğinin tahminidir. Literatürde müşteri davranışları incelemesi olarak adlandırılan müşterilerin hareketlerinden, ürünlere verilen tepkilerden, harcamalarından analiz yaparak daha iyi satış yapılabilir mi sorusunun cevaplanması vardır [5].

Literatürde costumer behaivoral olarak geçmekte olan bu kavramı ve bu kavram için gerekli analizlerin nasıl toplanıldığından bahsetmemiz gerekirse müşterilerin arama geçmişlerinin analizleri, müşterilerin markalar üzerindeki davranışları, müşterilerin ürünlere ulaşmak için kullandıkları reklamlar ve methodlar, reklam ve methodların kullanıcılara sunuluş biçimleri (bir e ticaret sitesi ise uygulama ve sitenin kullanıcı dostluğu, reklamların kullanıcılara nasıl ulaştırıldığı, ürün dizilimleri ve kategorilendirmeleri), müşterilerin ürünler üzerindeki geçmiş akışı ve sistemde kalma süreleri, geçmişteki yapılan alışverişler ve alışveriş kategorileri, sistemi kullandıkları zaman aralıkları (günün hangi saatlerinde, hangi günlerde ve bu günün özel bir gün olup olmadığının kontrolü), müşterilerin demografik bilgilerinin analiz edilmesi bu analizlerden tahminlenmesinde kullanılan değişkenlerdir[5]. Bu değişkenlerin ve davranışların doğru analiz edilmesi sonucu gerekli yorumların yapılması ile birlikte potansiyel olarak alma ihtimali yüksek kullanıcılara daha verimli ve etkili bir şekilde satışlarımızı gerçekleştirebilir[5]. Müşterileri ve ürünleri doğru kategorize edebilmek satışımızda ve müşterinin ihtiyacı olan ürüne erişiminde büyük bir etken olmaktadır. Uygun bir gelir dağılımına ya da üründen farklı tipteki bir ürünü o müşteriye reklam bildirim veya herhangi bir yola başvurarak müşteriye ilerletmemiz ilgisini çekmeyecektir. Bu yüzden müşterilerimizi ve ürünlerimizi iyi analiz edip doğru bir kategorilendirme vasıtası ile ürünlerimizi müşterilerimize daha kolay ve daha etkili sunabiliriz.

Bu anlatımların sonucunda yapılan bir literatür taramasından ve geçmişteki yapılan araştırmalar sonucundan kısaca bashetmemiz gerekirse Culotta ve Cutler, müşterilerin marka algılarını, sosyal medya verileri kullanarak, zahmetsiz ve maliyetsiz bir yöntemle analiz etmişlerdir. Veri madenciliği tekniklerinin kullanıldığı çalışmada önceki çalışmalardan farklı olarak sosyal medya kullanıcılarının oluşturduğu içerikleri analiz etmek yerine, markanın müşterilerinin oluşturmuş oldukları ağlar analiz edilmiştir [6,7]. Müşteri davranışları insan davranışlarının da bir taklitidir. Bu sebepten aynı özellikleri sağlayan müşteriler gerek alışveriş platformlarında gerekse günlük hayatımızın vazgeçilmez bir yeri olan sosyal medya platformlarında karşılaştıkları marka reklamlarına göre belirli karakteristik özellikler gösterebilmektedirler. Bu sebeple aslında müşteri her yerde müşteridir.

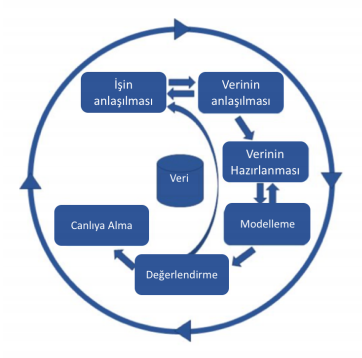
Yapılan bir literatür taramasında karşılaştığım bir çalışmadan bahsetmek gerekirse Kaneko ve Yada (2016), 1000 süpermarket müşterisine ait mağazada gezinme ve alışveriş işlemleri verilerini kullanarak, mağaza içerisindeki müşterilerin gezinmelerinin artması ile müşterilerin daha fazla alışveriş yaptıklarını tespit etmişlerdir. Veriler, müşterilerin alışveriş sepetine takılan bir RFID etiketi ile mağazadaki dolaşma verileri ve müşterilerin POS verilerini eşleştirerek elde edilmişti[6,8]. Daha öncesinde de bahsetmiş olunan ürün dizilimlerinin fiziksel olarak ve kanıtı olarak bu çalışmayı gösterebilmekteyiz. Aynı şekilde bu çalışma eğer ki bir e-ticaret paltformunda yapılıyor olsaydı buradaki raflar fiziksel bir raf değil, müşterilerin sistemleri kullandıkları arayüzler olacaktı.

**3. Makale Methodolojisi Ve Çalışma Amacının Belirlenmesi**

Çalışmamızda izlenecek methodoloji ve yöntem var olan verileri gerekli istatiksel modellere ve analizlere tutarak müşterilerin bir yapacakları alışverişlerin tutarını ve o işlemlerin ne kadar indirim ile gerçekleştirileceğinin belirlenmesi olacaktır. Bu çalışmanın gerçekleştirilebilmesi için özel kişisel verilerden arındırılarak insanlara gerçekteki verileri sunan Keggle platformu üzerinden üzerinde işlem yapabileceğimiz bir veri seti tedarik edilerek çalışmalara başlanmıştır. Bu çalışmalar python dilinin Sklearn Plotpy Matplotlib ve Seaborn kütüphaneleri kullanılarak bir notebook dosyası üzerinde gerçekleştirilmiş ve aynı şekilde çalışmanın anlaşılabilmesi için no-code yaklaşımını benimsemiş olan uçtan uca veri bilimi projeleri geliştirebilmemizi sağlayacak Knime Analitcs Platform kullanılarak geliştirmelere başlanmıştır.

**4. Methodolojinin Belirlenmesi**

Her veri bilimi sürecinde olduğu gibi bir metodoloji belirlenerek ve bu metodoloji adımları izlenerek iş geliştirme sürecine başlanmıştır. Metodolojilerin ama amacı bir veri bilimi projesine nasıl başlanacağı hangi adımların sırası ile devam edileceği ve sonucunda neler yapmamız gerektiğini belirten ve proje işleyişini yönlendiren yöntemlerdir. Semma KDD ve Crisp-DM olmak üzere bilinen ve kullanılan 3 adet metodoloji vardır. Güncel hayatta kullanışlılığı ve endüstriyel süreçlere daha rahat uyarlanabilmesi ve birçok endüstriyel süreç içerisinde uyumluluğu sebebiyle bu çalışmamız boyunca tercih ettiğimiz metodoloji CRSİP-DM (Cross-Industry Standard Process for Data Mining) metodolojisidir. Bu metodoloji detaylı bir şekilde açıklamak gerekirse 6 evreden oluşur ve bazı evreler arasında dönüşüm mevcuttur.



Şekil 1 CRISP-DM evreleri ve adım yönleri

**3.1.) İşin Anlaşılması (Business Understanding)**

Bu aşamada iş probleminin ve istenilenlerin anlaşılması evresidir. Problemin ne olduğunu, neden çıktığını, problem çözümü için iş işleyişinin anlaşılması, iş sürecinde ilgili verilerin tespiti, şayet problem çözülmesi olasılığını alarak hedeflerimizin belirlenmesi ve çözümü sonrası için başarı kriterleri nelerdir bunların analizi yapılır. Metodolojinin 1. Evresi olan bu evre en önemli ve hata yapılması durumunda en baştan başlanılması gereken evredir.

**3.2.) Verinin Anlaşılması (Data Understanting)**

Metodolojinin 2. evresi olan bu evrede iş tanımı ve hedeflerimiz doğrultusunda istenilen verilerin anlaşılmasına, gerekli görselleştirmelerin yapılarak verinin anlaşılmasını, eğer ki elimizde veri yoksa bu problem çözümü için uygun tipte veriler bularak bu verilerin anlaşılmasına nicelik istatistikleri çerçevesinde değerlendirilmesine bir sonraki aşamada işlenecek veri üzere gerekli tespitlerin yapılmasına dair bir işleyiştir.

**3.3.) Veri Hazırlanması (Data Preprosessing)**

Verinin hazırlanması aşamasından gelen verilerimizi ve işin anlaşılması sürecinden gelen isterler doğrultusunda verimizi gerekli bir şekilde işlemeye başlanılan aşamadır. Veri biliminde her problem her algoritma ile çözülmemektedir. Aynı şekilde her algoritmada aynı tipten verilerle çalışmamakta ver kendilerine özgü özellikler göstermektedir. Aynı amaca hizmet edilen verilerde ve algoritmalarda dahi aynı problem çözümü için farklı ön işlemeler yapılabilmektedir. Algoritma isterine göre verilerimizi görselleştirme doğrultusunda gelen bilgiler ışığında işleyerek verilerimizi algoritmalarımıza veririz.

**3.4.) Modelleme:**

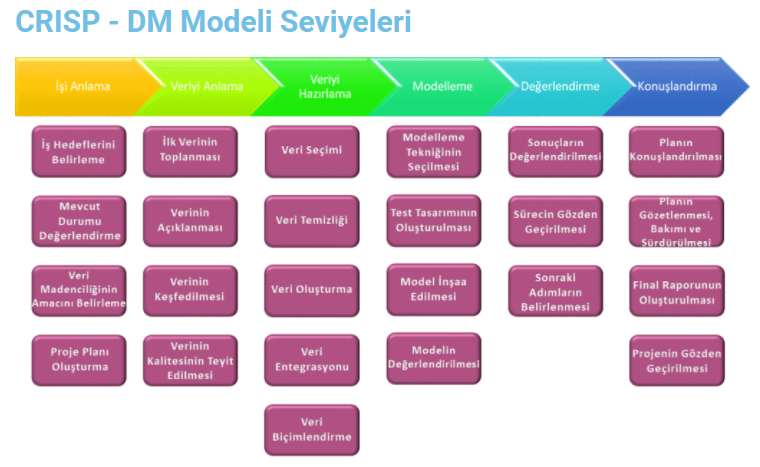
Ön işlemeden gelen veriler için bu aşamada gerekli makine öğrenmesi algoritmalarına vererek modellerimizin çıkartılması işlemidir. Gerekli kütüphaneler veya ortam üzerinden veriler öncesinde bir train ve test olarak bölünme yaşar. Train verileri ile sistem eğitilerek test kısmında ise daha önce algoritmanın hiç görmediği verilerin testi yapılır.

**3.5.) Değerlendirme (Evolution)**

Modelleme aşamasında üretilen modelin yorumlanacağı aşamadır. Birçok algoritmaya göre bu yorumlamalar farklılık göstermekte ve her problem türüne göre farklı bir yorumlanma biçimi barındırmaktadır. Aynı problem ve algoritmalar için dahi yorumlanmada değişik birçok metrik mevcuttur. Yorumlamalardan geçen ve daha öncesinde iş analizi kısmında belirttiğimiz değerlere göre kabul edilebilir değerler elde edebilirinse bir sonraki aşamada bu modeller canlıya alma (deploy) olarak adlandırılan aşamaya alınarak gündelik hayatta kullanılabilir bir biçime dönüştürülecektir.

**3.6.) Canlıya Alma : (Deploy)**

Alınan modellerin kabul edilebilir olması durumunda canlı bir hayata uygulanması ve gerekli dönüşümler yapılarak sistemlere aktarılması işi olarak düşünülebilir. Bu bir web programı için ise gerekli backend işlemleri için arka planda bağlanabilir, satış tahminlemeye dair bir otomasyonsa gerekli uygulamaya eklenerek kullanıcıların kullanımına sunulabilir.

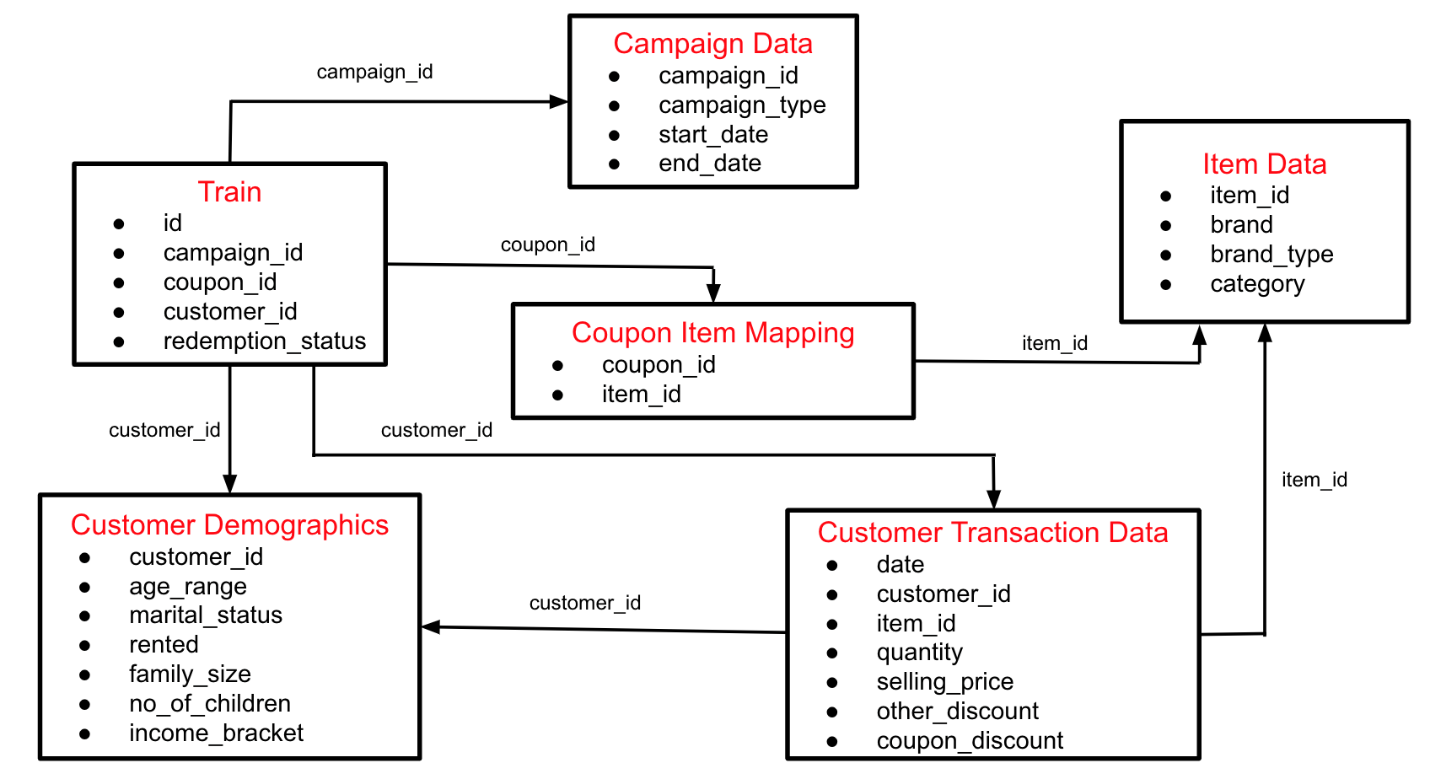
****

Şekil 2 CRİSP-DM aşama bazlı açıklayıcı şablon

**5. Verilerin Anlaşılması Aşaması**

CRİSP-DM metodolojisinin 2. basamağı olarak bilinen bu aşamada işe başlanılabilmesi için bir veri kümesi toplanması gerekliliği mevcuttur. Veri bilimi ve makine öğrenmeleri projelerinde kullanıcılara kişisel verilerinden olabildiğince arındırılmış ve var olan iş problemlerine uygun olarak bir sürü veri seti içerisinde barındıran Keggle platformu üzerinden bir veri seti tedariki gerçekleştirilmiştir.

Veri seti seçiminde makalenin 2. başlık altında belirtildiği üzere müşteri davranışlarının analizi ve satış ilişkisi olarak incelemek istediğimiz bu çalışmada veri seti kolonlarımızın müşterilerin demografik özelliklerini barındıran, eğer mümkünse ürünler üzerinde gerçekleştirdikleri davranışların olduğu niteliklere sahip olan, ürün özelliklerinin de içerisinde barındığı bir tablo tipi ile çalışılması gerekmektedir. Bu çalışma için Keggle platformu üzerinde “Predicting\_Cupon\_Redemption\_Pca” isimli veri setinin çalışma amacının doğrultusunda uygun olacağı düşünülerek seçilmiş bulunmaktadır.



Şekil 3 Tedarik edilen veri seti hakkında tablolar arası ilişkiler şablonu

Veri setimizin tablo analizlerini yapmamız gerekirse costumer\_demograpich isimli tablomuzda müşterilerin kişisel özelliklerinin olduğu (müşterilerin demografik özellikleri) costumer\_transaction tablosunda müşterilerimizin yaptıkları işlemler ve bu işlemleri gerçekleştirdikleri zaman dilimi, tane adeti, indirim (bu alanda indirim para bazında verilmiş bulunmaktadır) gibi özellikleri barındıran, item\_data tablosunda ise ürün özelliklerimizi barındıran, cupon\_item\_mapping tablosunda ise ürünler ile kuponların eşleşmelerini içeren ve campaing\_data isimli tabloda ise kampanya süreleri ve bu kampanyaların özelliklerini belirten bir veri seti olarak karşımıza çıkmaktadır.

Veri analizinin ilk aşaması olarak veri seti tedarikinden ve bu veri setine genel bir bakıştan sonra bu projelerin gerekli geliştirme ortamları içerisine dahil edilerek sırası ile değişkenlerin açıklanması, keşifçi veri analizi (ead) ve Öznitelik mühendisliği (feature engineering) ile ilerleyerek devam edilecektir.

Çalışma ortamı için gerekli kütüphaneler tedarik edilerek geliştirme ortamına eklenmesi sağlandıktan sonra bu kütüphaneleri kullanabilmemiz için python editörümüz üzerinden çağırmamız gerekmektedir. proje işleyişi boyunca, pythonda sklearn, seaborn, matplotlib kütüphaneleri tercih edilerek çalışma gerçekleştirilmiştir.

|  |
| --- |
| **import** **numpy** **as** **np**  **import** **pandas** **as** **pd**  """ ----------------------- Görselleştirme Kütüphaneleri ----------------------- """  **import** **matplotlib.pyplot** **as** **plt**  **import** **seaborn** **as** **sns**  sns.set()  plt.style.use("seaborn-whitegrid")  """ ------------------------- Ön İşleme Kütüphaneleri ------------------------- """  **from** **sklearn.preprocessing** **import** LabelEncoder  **from** **sklearn.preprocessing** **import** OneHotEncoder  **from** **sklearn.preprocessing** **import** StandardScaler  **from** **sklearn.impute** **import** SimpleImputer  **from** **sklearn.model\_selection** **import** train\_test\_split  **from** **sklearn.model\_selection** **import** cross\_val\_score  """ ---------------------- Model Değerlendirme Kütüphaneleri ---------------------- """  **from** **sklearn.metrics** **import** r2\_score  **import** **statsmodels.api** **as** **sm**  """ ------------------------------ Model Kütüphaneleri ------------------------------ """  **from** **sklearn.linear\_model** **import** LinearRegression  **from** **sklearn.ensemble** **import** RandomForestRegressor  **from** **lightgbm import** LGBMRegressor  **from** **sklearn.tree** **import** DecisionTreeRegressor  **import** **xgboost** **as** **xg**  **from** **sklearn.ensemble** **import** GradientBoostingRegressor  **from** **regressors** **import** stats  """ ------------------------------ Hiperparametre Optimizasyonu ------------------------------ """  **from** **sklearn.model\_selection** **import** GridSearchCV  """ ------------------------------ Model Dışa Aktarma ------------------------------ """  **import** **pickle** # yapılan modelleri kaydetmek için kullanılmaktadır |

Gerekli kütüphanelerin yüklenmesi sağlandıktan sonra kullanılacak veri setlerinin projeye dahil edilmesi için veri setlerinin manipülasyonu ve üzerinde işlemler yapmamızı sağlayan pandas kütüphanesinin altından read\_csv() methodu çağırılarak veri setlerimiz çalışma ortamımıza eklenmiştir.

|  |
| --- |
| kampanya\_data=pd.read\_csv("campaign\_data.csv")  kupon\_item\_eslesmesi=pd.read\_csv("coupon\_item\_mapping.csv")  musteri\_demografik=pd.read\_csv("customer\_demographics.csv")  musteri\_islemleri=pd.read\_csv("customer\_transaction\_data.csv")  urun\_verileri=pd.read\_csv("item\_data.csv") |

Kullanılan dosyalar CSV uzantılı olduğundan ötürü read\_csv methodu kullanılmıştır. Veri setinin boyutsal olarak anlaşılabilmesi açısından DataFrame objelerinin bir methodu olarak shape fonksiyonu kullanılarak boyutsal manada bilgiler aşağıdaki gibi elde edilmiştir.

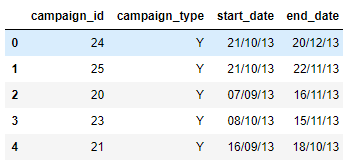
|  |  |
| --- | --- |
| **print**("Kampanya Datasının boyutu : {0} \nKupon\_item Datasının boyutu : {1} \nMusteri Demografik Datasının boyutu : {2} \nMusteri İşlem Datasının boyutu : {3}\nÜrün Bilgileri Datasının boyutu : {4} \n".format(kampanya\_data.shape, kupon\_item\_eşleşmesi.shape,musteri\_demografik.shape,musteri\_islemleri.shape,urun\_verileri.shape))   |  | | --- | | >>>  Kampanya Datasının boyutu : (28, 4)  Kupon\_item Datasının boyutu : (92663, 2)  Musteri Demografik Datasının boyutu : (760, 7)  Musteri İşlem Datasının boyutu : (1324566, 7)  Ürün Bilgileri Datasının boyutu : (74066, 4) | |

**5.1.) Veri Setlerinin Anlaşılması Ve Görselleştirmesi:**

Bu aşamada veri setlerimizi incelerken değişkenlerin tanımlanması, özelliklerinin incelenmesi ve keşifçi veri analizi işlemlerini tek tek tablosal bazda gerçekleştirip gerekli tablo için ilgili başlığın altından verilerine erişebilirsiniz.

**5.1.1.) Campaign\_Data Tablosu**

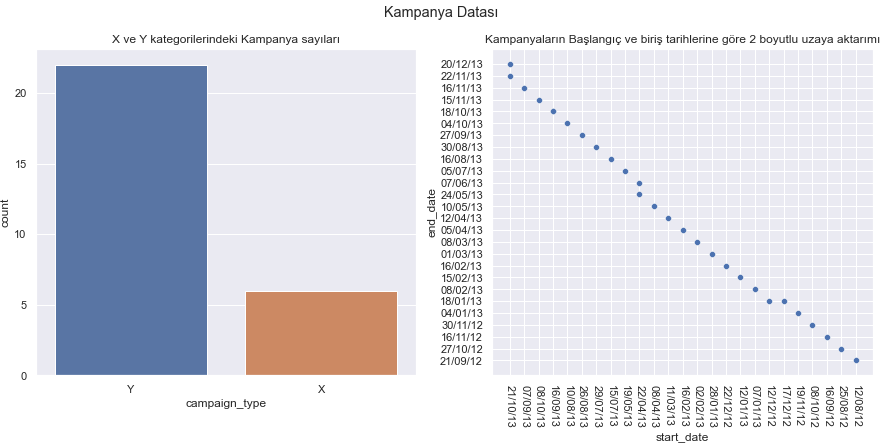
Campaign\_data kampanyalar hakkında bilgi vermektedir. Hangi tarihler arasında hangi tipte kampanyaların olduğunu vermektedir. Boş veri barındırmamaktadır. Bunula birlikte campaing\_id haricindeki tüm veriler kategorik bir veridir. 28 kayıt mevcuttur.



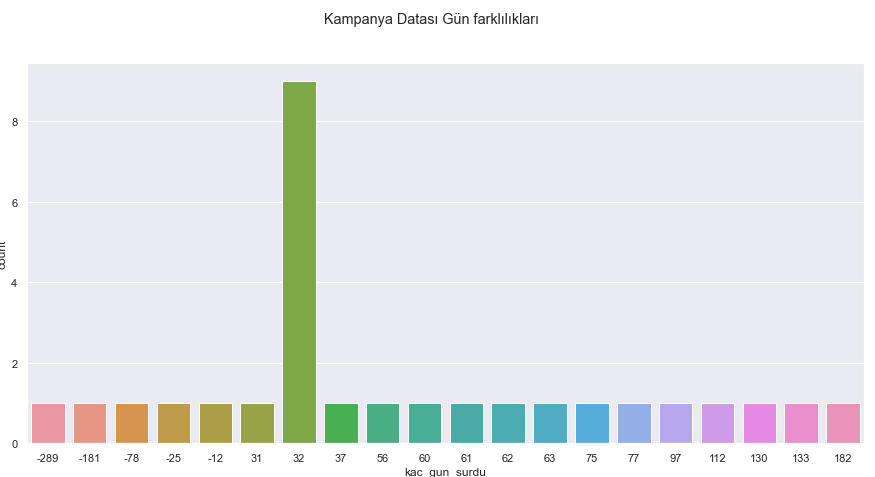
**campaign\_id:** kampanyalarımızın benzersiz kılan bir sütundur.

**campaign\_type:** kampanya verilerimizin tipleri olarak 2 adet değişkenimiz mevcuttur bunlar x ve y değişkenleridir.

**start\_date ve end\_date:** Kampanya başlangıç ve bitiş tarihlerini vermektedir.



Şekil 4 Kampanya Tipleri ve Kampanya Tarihleri Tabloları

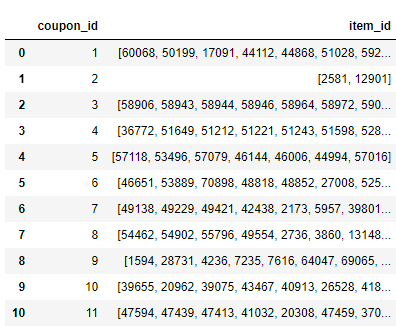


Şekil 5 Kampanyaların Geçerlilik süreleri

Elde edilen grafikler neticesinde bir yorumlama yapmak gerekirse x kampanyasına ait 6, y kampanyasına ait 22 adet verimiz bulunmaktadır. Aynı tarihlerde başlayan iki adet kampanyamız ve aynı tarihlerde biten 1 adet kampanyamız bulunmakta birbirleri ile aynı tarihe gelen birden çok kampanyamızda mevcut bulunmaktadır. Değişken tiplerimiz 1 adet numerik (id) 3 adette kategorik (başlangıç ve bitiş tarihleri, kampanya tipleri) olarak görülmektedir.

**5.1.2.) Coupon\_item\_mapping Tablosu**

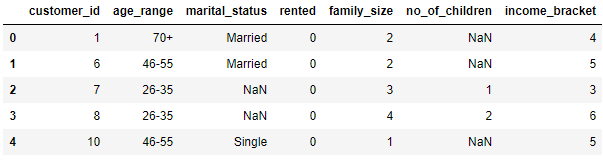
Bu tabloda coupon\_id lerimiz ve bu idlerimiz ile eşleşen item\_id lerimiz mevcuttur. Yani hangi ürüne hangi kampanyların eşleştiğini gösteren bir tablodur. Toplamda 92663 veri barındıran bu tablomuzda boş veri içermemektedir. Çalıştırılacak ufak bir groupby ile elde edebileceğimiz verilerden basetmemiz gerekirse bir item ile eşleşen birden fazla kupon ve bir kuponun birden fazla veri üzerinde kullanıldığının tespiti yapılmıştır. Veri setinden ilk 11 örneği aşağıdaki tabloda verilmiştir.



Şekil 6 Kuponların ürün\_id leri ile olan eşleşmeleri

**5.1.3.) Customer\_demographics Tablosu**

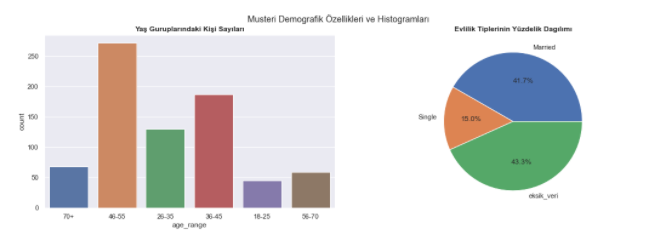
Bu tablonun içerisindeki veriler sistemdeki müşterilerin demografik özelliklerini vermektedir. 760 adet müşterimizin Özelliklerin ne anlama geldiğinin açıklamaları aşağıda mevcut bulunmakta ve tablomuzun anlaşılabilmesi için ilk 5 satırı gözükmektedir.



Şekil 7 customer\_demographics tablosundan ilk 5 satırın alınması

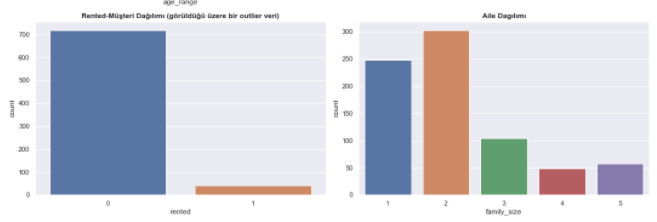
* **Costumer\_id:** Budeğişken müşterileri benzersiz kılan bir anahtardır.
* **Age\_Range:** Müşterilerin yaş aralıklarının bir gurup etrafında toplanmasıdır. belirli yaş aralılarının belirli guruplara bölündüğü gözlemlenmiştir. Bu bölümlerin Label Encoding yöntemi ile sıralandırılması gerekrir. Veriler ordinal olarak sıralanabildiği için bu yöntemi seçmemiz gerekmektedir.
* **Marital\_Status:** Azımsanmayacak kadar boşluklu değerler içeriyor. Bunu doldurmanın bazı yolları mevcuttur. Eğerki yeterli miktarda doldurma gerçekleşmezse bu sütunu atmamız gerekecektir. (boş değerlerle çalışamayan algoritmalarımız var bu sebeple çıkarmamız gerekecektir.) 329 adet boş veri vardır.
* **Rented:** Müşterilerin kirada mı yoksa ev sahibimi olduğunu belirten bir değişkendir.Dengesiz bir dağılım göstermiş bulunmaktadır (unbalanced) . Bu sebepten dolayı biz bunuda çıkarmalıyız. alttaki grafikte 0 ve 1 in ne kadar dengesiz oldugu gösterilmiştir.
* **Family\_size**: degişkeninde + degerli ifadeler bulunmaktadır. 5 ve 5 ten yukarısının bir gurup altında toplandığı belirtilmiştir. bu + yı gerekli lambda fonksiyonu ile sildirmemiz gerekecektir. Bu artılı ifadeler aile boyunun sayısı hakkında bilgi vermektedir.
* **No\_of\_childiren:** Müşterilerimizin çocuk sayılarının belirtildiği değişkendir. Degişkeninde + degerli ifadeler var bu + yı gerekli lambda fonksiyonu ile sildirmemiz gerekecektir ve bu + ile 3 ve üstü olan gurupları tek bir değişken altında toplanmıştır. ve boş değerleri belirli bir mantıkla dolduracağız. 538 adet boş verimiz vardır.
* **İncome\_bracket:** müşterilerimizin sosyodemografik gelir seviyeleri hakkında fikir sahibi olmamızı sağlayan veridir.

Yapılan ön analiz sonucu sistemde marital status durumu için 329 adet boş değişken ve no\_of\_children için ise 538 adet boş değişkenimiz mevcuttur.

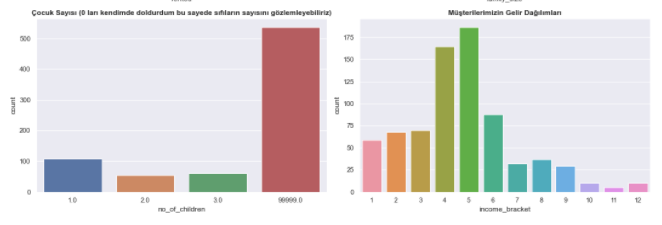
EDA için veriler hakkında temel ve etkili bilgi verebilecek bir yöntem olarak histogram grafiklerine bakılması bu aşamada bize yeterince bilgi verecektir.

Şekil 8 Yaş gurupları ve evlilik durumlarının dağılımı

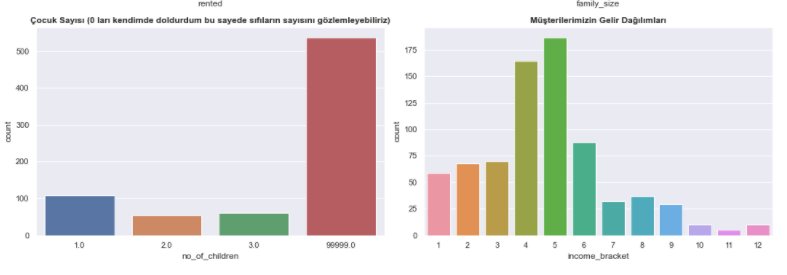
Bu grafikler öncesinde histogram grafiklerini çıkartırken boş değerlerin sayısının gözlemlenebilmesi adına sklearn.preproesessing kütüphanesinden bir simple\_imputer objesi oluşturularak boş değerler eğitilip sabit bir değerle doldurulmuştur. Doldurmalar sonucu boş değerlerin dağılımı hem eksik\_veri isimli değişken ile evlilik statüsünde hemde 9999 olarak çocuk sayısında gözlemlenmiş bulunmaktadır. Yukarıdaki evlilik tiplerinin yüzdelik dağılımı grafiğinde gözüktüğü üzere %43.3 oranında boş verimiz mevcuttur. Modelleme aşamasına geçmeden önce belirtmek gerekirse her algoritmaların bazı hassas oldukları alanlar mevcuttur. Bu doğrultuda içerisinde boş ifadeler içeren değişkenlerle çalışılmama ihtimali olan algoritmaları kullanacağımız zaman bu boş hücre değerleri bize sıkıntı çıkartacaktır. Bu sebepten ötürü tablodaki diğer değişkenler yorumlanarak bu değeler doldurulmaya çalışılacaktır.



Şekil 9 Rented ve aile boyutu histogramları

Yukarıdaki rented tablosuna bakıldığı zaman bir değişkenimizin dengesiz dağılım gösterdiği ve tabloda 1 değerinin çok ağır bastığı görülmektedir. İstatiksel açıdan açıklamak gerekirse böyle bir durumda bu değişkeni sistemden elememiz gerekebilir. Bu sebepten ötürü ilerleyen aşamalarda tekrar incelenerek üzerinde gerekli manipülasyonlar yapılacaktır. 1 değeri sistemi dengesiz yapmaktadır.

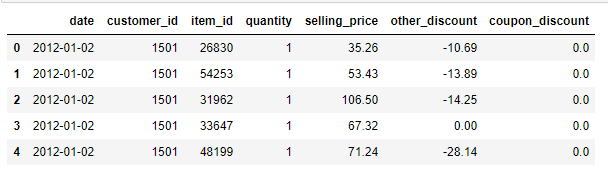
Aşağıdaki grafiklere öncesinde histogram grafiklerini çıkartırken boş değerlerin sayısının gözlemlenebilmesi adına sklearn.preproesessing kütüphanesinden bir simple imputer objesi oluşturularak boş değerler eğitilip sabit bir değerle doldurulmuştur. No\_of\_children değişkeninin boş değerlerine 99999 sayısı atanarak ne kadar bir boşluk olduğunu ve frekansa nasıl bir etkisi olduğunu bu aşamada gözlemlemiş bulunduk. Ön işleme (preprocessing) aşamasında tablodaki diğer değişkenler kullanılarak bu değişkenler doldurulmaya çalışılacaktır.



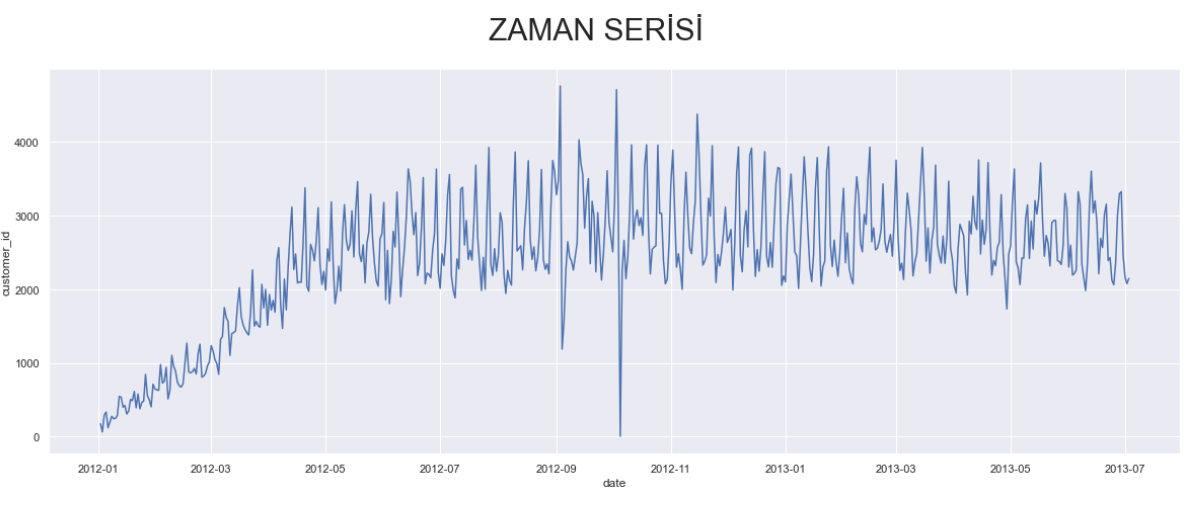
Şekil 10 Boş değerleri olan çocuk sayısı değişkeni ve gelir seviyelerinin dağılımları

**5.1.4.) Customer\_Transaction Tablosu**

Tablonun içeriği müşterilerimizin gerçekleştirdiği işlemlerin özellikleri belirtilmiş bulunmaktadır. Özellikleri detaylı bir şekilde aşağıda belirtilmiş bulunmaktadır.

* **Date**: İşlemin gerçekleşme zamanını verir. (yyyy-mm-dd formatında)
* **Costumer\_id:** müşteri demografik tablosu içinbenzersiz değer taşıyan bu değerimiz bu tabloda hangi müşterinin hangi ürün üzerinde işlem uyguladığını belirtmekte olup bir yapancı anahtardır.
* **İtem\_id:** costumer\_id deki aynı özellikleri gösterip item\_dataset için geçerlidir.
* **Quantity:** İşlem içerisinde belirtilen üründen kaç adet satıldığını belirten değişkenimizdir.
* **Selling\_price:** İşlemin gerçekleştiği fiyat değişkenidir. Daha detaylı bir şekilde açıklamamız gerekirse bu değişkenimiz işlem gerçekleşirken uygulanmış olan tüm indirimler sonucu fatura fiyatı olarak adlandırabiliriz.
* **Other\_discount:** Ürünün kupon kullanılmadan girdiği indirim miktarıdır. Bu değişkenimiz negatif veriler içermekle birlikte indirimin parasal nicelikten karşılığı manasına gelmektedir.
* **Coupon\_discount:** Ürünlerin satın alımında kullanılan kupon ve kampanyalar çerçevesinde müşteriye uygulanan indirim miktarıdır. Bu değişkenimiz negatif veriler içermekle birlikte indirimin parasal nicelikten karşılığı manasına gelmektedir.

Şekil 11 Müşteri işlemleri tablosundan ilk 5 satır örneği

Gözüktüğü üzere zaman eksenini sıralı (ordinal) bir şekilde zamansal düzlemde baktığımız kadarı ile belirli tarihler arası lineer artış göstermektedir. Zaman analizinde tarihler bazında o gün ne kadar işlem gerçekleştiği ile ilgili bir incelemedir. Bu açısı dan bakılınca zaman değişkenimizin satışlar bazında tarihle ilişkisinin olabileceği kurgulanmıştır. Öte yandan belirli bir değerden sonra çok yüksek oranda bir dalgalanma içermekte ve sistemimizi yanıltabilecek etken verebilir.

Şekil 12 Günlük olarak satin alım sayıları

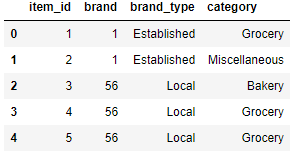
Şekildeki grafiğe bakılacak olursa genellikle istatiksel açıdan düzenli bir dağılım göstermemekle birlikte bazı zaman dilimlerin tepe nokralarında ve bazı zamanlarda ise alt noktalara vardığı gözlemlenmektedir. Bu sebepten ötürü hem işleyiş boyunca tarih değişkenlerimizi günün ayları haftanın günleri ve bazında yapılacaktır.



Şekil 13 Toplam indirim analizi

**5.1.5.) İtem\_Data Tablosu**

Bu tablomuz ürünlerin özelliklerinin tutulduğu bir tablodur. Tablomuzda toplamda 74066 adet ürünümüz mevcuttur

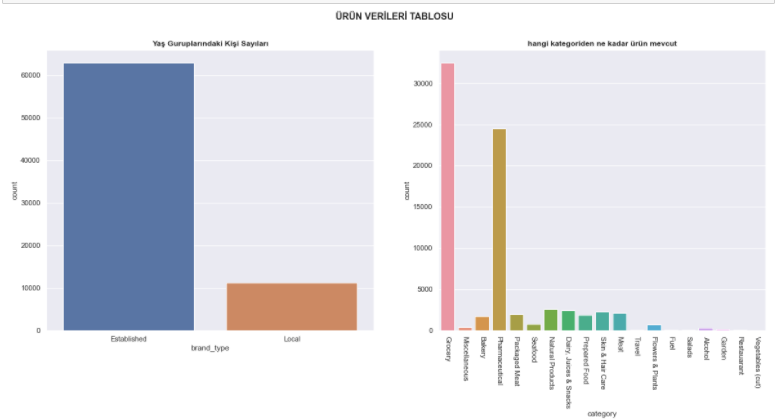
****

**İtem\_id:** ürünlerimizin benzersiz numarası olarak tanımlanmaktadır.

**Brand:** ürünlerin markasını vermektedir.

**Brand\_type:** ürünün tipini vermektedir.

**Category:** ürünün kategorisini vermektedir.



Şekil 14 marke tipleri ve ürün kategorileri histogram grafikleri

şekildeki gözüktüğü üzere dengesiz (inbalanced) bir dağılım söz konusudur.

**6. Ön İşleme (Preprocessing) ve Özellik Mühendisliği**

Keşifçi veri analizinde verilerimizin bizlere neler anlattığını ve verimiz hakkında gerekli tespitlerimizde yaptığımıza göre bu tespitler çerçevesinde verimizi vereceğimiz makine öğrenmesi algoritmasına göre gerekli manipülasyonları yaparak ve yeni özellikler çıkartarak (feature engineering) sisteme vereceğiz. Tek tek tablo bazında ön işleme işlemlerini incelemek gerekmektedir.

**6.1.) Campaign\_Data Tablosu:**

Değişkenlerimizi bu aşamada hatırlamak gerekirse campaign\_idisimliütünün kampanyalarımızı benzersiz yapacak bir tanımlayıcı değişkeni olduğuna, campaign\_typeisex ve y olmak üzere 2 adet değişkenden oluşan ve kampanyalarımızın tipini belirleyen bir değişken, start\_date ve end\_date ise kampanyalarımız için başlangıç ve bitiş aralığı verilmesi işidir.

campaign\_idisimli değişkenimizde herhangi bir işlem yapmamız gerekmemektedir. Bu değişken ileride birleştirme işlemlerinde bizlere anahtar sağlayacaktır.campaign\_type ise 2 tipten oluştuğu için sklearn kütüphanesinin altından preprosessing.LabelEncoder()objesi çağırılarak bir değişkenler kategorik verilerden numerik verilere çevrilecektir.

|  |
| --- |
| le=LabelEncoder()  kampanya\_data["campaign\_type"]=le.fit\_transform(kampanya\_data["campaign\_type"]) |

Start\_date ve end\_date değişkenleri için python veri tipi dönüşümlerinden bu değişkenleri zaman değişkenine çevirmemiz gerekmektedir.

|  |
| --- |
| # Kategorik olan tarhi değişkenlerini tarih cinsine çevirmek  kampanya\_data['start\_date'] = pd.to\_datetime(kampanya\_data['start\_date'])  kampanya\_data['end\_date'] = pd.to\_datetime(kampanya\_data['end\_date'])  # Kampanya geçerlilik süreleri hesaplama  kampanya\_data['kac\_gun\_surdu'] = kampanya\_data['end\_date'] - kampanya\_data['start\_date']  kampanya\_data["kac\_gun\_surdu"]=kampanya\_data["kac\_gun\_surdu"].apply(lambda x: x.days) |

Yapılan incelemeler sonucu şekil 4 te de görüldüğü üzere 32 gün süren 8 adet kampanyamız bulunmaktadır.

Bu kampanyalarımız ise 25, 21, 22, 19, 17, 16, 12, 4, 2 numaralı kampanyalardır.

### **6.2) Coupon\_item\_mapping Tablosu:**

Bu veri seti ise ürünler ve bu ürünlerle eşleşen kuponların bulunduğu bir eşleştirme tablosudur. Tabloları birbirlerine bağlamamız gerekirse bu değişken üzerinden gerekli bağlama işlemleri gerçekleştirilecektir. Bir ön işlemeye ihtiyaç yoktur.

**6.3) Customer\_demographics Tablosu:**

Müşteri demografik özelliklerini içeren bu tabloda gerekli önişlemeler aşağıda gösterildiği gibi gerçekleştirilmiştir.

* **Age\_Range:** kullanıcıların yaş değerleri belirli kuruplandırmalar yapılarak aralıklara bölünmüştür. Yaş verisi kategorik olarak burada sınıflandırılsa da gerçek hayat üzerinden düşünüldüğüne bu bir sayısal veri olup sıralandırılabilen (ordinal) bir veri tipidir. Örnek vermek gerekirse 70 ve üzserindeki yaş gurubundaki insanlar 46-55 yaş aralığındaki insanlardan yaş olarak büyüktür. Bu guruplandırmaların analizi neticesinde sklearn kütüphanesinin preprocessing sınıfının altında bulunan labelencoder sınıfından bir nesne türetilerek gerekli olan label encoding işlemini bu aşamada gerçekleştirilecektir.

|  |
| --- |
| encoder=LabelEncoder()  #Label Encoding Evlilik Statüsü --- 0 ise Bekar Ve 1 İse Evli  musteri\_demografik[["marital\_status"]]=encoder.fit\_transform(musteri\_demografik[["marital\_status"]])  # Label Encoder age\_range ... 18-25 ise 0, 26-35 ise 1, 36-45 ise 2, 46-55 ise 3, 56-70 ise 4 ve 70+ ise 5 olarak çevirecektir.  musteri\_demografik[["age\_range"]] = encoder.fit\_transform(musteri\_demografik[["age\_range"]]) |

* **Marital\_Status:** bu değişken bir önceki incelememizde toplam değişken sütununun %43.3 ü kadarlık bir kısmı boş değer içermektedir. azımsanmayacak kadar boşluklu değerler içermesi sebebi ile bu sütunları bağzı mantıksal çerçeveler ve diğer sütunlardan yardım alarak doldurmaya çalışacağız. Bu değişkeni doldurabilmenin bağzı yolları mevcuttur. Eğerki yeterli miktarda doldurma gerçekleşmezse bu sütunu hem algoritmalarımızın yanlış bir değer tahmini yapmasına sebebiyet verebileceğinden veya çalışmasını engelleyebileceğinden ötürü çıkartmamız gerekecektir. Doldurma işlemi için herhangi bir istatiksel model veya süreç tercih edilmemiş olup kural tabanına göre bir doldurdma işlemi gerçekleştirdik. Kuralları oluşturmamızda bize yardımcı olan no-of-children ile family-size sütunları baz alınarak bir doldurma işlemi gerçekleştirilmiştir.

|  |
| --- |
| #aile sayısı 1 olan müştrilerin hepsi bekardır hem çocuk yok hem eş yok  musteri\_demografik.loc[pd.isnull(musteri\_demografik.marital\_status) & (musteri\_demografik.family\_size == 1),'marital\_status'] = 'Single' # bu şekilde 183 kişi dolduruldu  #aile sayısı - çocuk sayısı = 1 koşulunu sağlayan müştrilerin hespi bekardır musteri\_demografik.loc[(musteri\_demografik.family\_size - musteri\_demografik.no\_of\_children == 1) & pd.isnull(musteri\_demografik.marital\_status),'marital\_status'] = 'Single' # 33 deger dolduruldu  # bir kişi evli ve aile sayısı iki ise çocuk sayısı 0 dır . mantıklı  musteri\_demografik.loc[pd.isnull(musteri\_demografik.marital\_status) & (pd.isnull  (musteri\_demografik.no\_of\_children))& (musteri\_demografik.family\_size ==2),'marital\_status'] = 'Married' |

* **Rented:** düzgün bir dağılım göstermemiş olan bu dağılımın değişkenleri bir sayı olup bu sebepten dolayı herhangi bir preprocessing yapmamıza ihtiyaç kalmamaktadır.
* **Family\_Size:** Aile boyutunu belirten bu değişkende 5 ve 5 in üstündeki boyuta sahip aileler 5+ olarak bir kategoride toplanmış bulunmaktadır. Öncelikle biz bu artı ifadesini kaldırıp ifadeyi kategorik bir veriden sayısal bir veriye çevimemiz gerekmektedir.

|  |
| --- |
| musteri\_demografik['family\_size'] = musteri\_demografik.family\_size.apply(**lambda** x: int(re.sub('\+','',x))) |

* **No\_Of\_Childiren**: Çocuk sayısını belirten bu değişkende 3 ve 3 ün üzerindeki çocuk sayıları için 3+ ifadesi kullanılmıştır. Bunula birlikte içerisinde yüksek oranda boş değer içermektedir. Bu boş değerlerin doldurulması için birçok istatiksel yöntem bulunsa dahi kural tabanlı doldurmak en doğru karar olacaktır. Doldurma işlemleri için evlilik durumu ve aile sayısı değişkenlerinden faydalanılacaktır.

|  |
| --- |
| # ilk aşamada + olan değişken silinerek işleyişe başlandı  musteri\_demografik['no\_of\_children'] = musteri\_demografik.no\_of\_children.apply(**lambda** x: int(re.sub('\+'  ,'',x)) **if** pd.notna(x) **else** x)  #evli insanların aile sayısı iki ise ve ailede iki kişi varsa çocuk yoktur  musteri\_demografik.loc[pd.isnull(musteri\_demografik.no\_of\_children) & (musteri\_demografik.marital\_status == 'Married') & (musteri\_demografik.family\_size == 2),'no\_of\_children'] = 0  #Aile sayısı 1 olan müşteriler ise çocuk sahibi olmayacagından ötürü çcouk sayısı 0 olacaktır  musteri\_demografik.loc[pd.isnull(musteri\_demografik.no\_of\_children) & (musteri\_demografik.family\_size == 1), 'no\_of\_children'] = 0 |

* **İncome\_Bracket:** Gelir seviyesini gösteren bu değişkenimiz için herhangi bir önişleme yapma ihtiyacımız yoktur.

**6.4.) Customer\_Transaction Tablosu:**

Müşterilerimizin zaman bazında hangi ürünler üzerinde ne kadar miktarda ve ne kadar indirimler ile gerçekleştirildiği gibi birtakım bilgilerini içeren bu tablomuzda değişken bazında yapılacak işlemleri incelememiz gerekirse

* **Date:** yyyy-mm-dd(yıl-ay-gün) formatında olanbu değişkenimiz date olarak zamana çevrilecektir. Bununla birlikte zaman analizindeki incelemeler sonucu müşterilerin zaman dilimine göre alışveriş alışkanlıkları değişebilme ihtimaline karşılık özellik mühendisliği kullanılarak bu zaman diliminden yılın ayları, ayın günleri ve haftanın günleri olmak üzere 3 adet yeni özellik çıkartmamız gerekmektedir.

|  |
| --- |
| # Zaman Değişkenimiz Zamana Çevrildi  musteri\_islemleri['date'] = pd.to\_datetime(musteri\_islemleri['date'])  #gerekli özellik mühendisliği yapıldı  musteri\_islemleri['day\_of\_mounth'] = musteri\_islemleri["date"].apply(lambda x: x.day)  musteri\_islemleri['day\_of\_week'] = musteri\_islemleri["date"].apply(lambda x: x.weekday())  musteri\_islemleri['yer\_of\_mounth'] = musteri\_islemleri["date"].apply(lambda x: x.month) |

* **Selling\_Price:** Değişkenimiz işlemin toplam ne kadara geldiğini göstermektedir. Bu fiyat değişkeni toplamda diğer indirimler ve kullanılan kupon indirimlerin sonucunda işlemin ne kadar tuttuğunu belirtmektedir. Hedef Değişkenimiz ve aynı zamanda tahmin etmeye çalıştığımız değer olan müşterilerin yüzdelik üzerinden ne kadar indirim ile ürünlerin alınacağının hesaplanması yapılırken bu değişkenimizle birlikte indirim değişkenleri kullanılarak indirimsiz halinin ne kadara geldiğini hesaplamamız gerekecektir.
* **Other\_discount:** İşlemde bir indirim varsa bu alanda gözükmektedir. Verilerimiz eksi veriler olup özellik mühendisliğinde bu değerleri pozitif sayıya çevirip hem indirimsiz halini bulmamıza hem de indirim oranı bulmamıza yardımcı olacak bir değişkendir.
* **Coupon\_discount :** işlemlerimizde eğer ki bir kupon kullanıldı ise veya kampanyadan yararlanıldı ise bu değişkende para cinsinden ne kadar indirime girdiği belirtilmiştir.

Belirtilen son 3 değişken çerçevesinde öznitelik mühendisliği ve ön işlemeler aşağıdaki şekilde uygulanmıştır.

|  |
| --- |
| # ürünler indirime girmeseydi işlem ücretinin ne kadar olacagını hesaplayalım  musteri\_islemleri["selling\_price\_indirimsiz"] = (musteri\_islemleri.selling\_price musteri\_islemleri.  other\_discount-musteri\_islemleri.coupon\_discount)  # İşlemler gerçekleştiginde toplam ne kadarlik bir indirim oldugunu hesaplayalım  musteri\_islemleri["toplam\_indirim"]= abs(musteri\_islemleri.other\_discount+musteri\_islemleri.coupon\_discount)  # şimdi sırada hedef kolonumuzu belirtmemiz gerekiyor. Hedefimiz müşterilerimizin bu işlemi yüzde kaç indirimil gerçekleştireceğinin hesaplanmasıdır.  musteri\_islemleri["yuzde\_kac\_indirim"]=round((1(musteri\_islemleri["selling\_price"]/musteri\_islemleri["selling\_price\_indirimsiz"]))\*100,3) |

Bu aşamada ürünlerin indirimsiz durumda ne kadara satılacağının hesaplanması ve eksi olan indirim değişkenlerinin regresyon üzerinde negatif bir etki oluşturabileceğinden kaynaklı olarak gerekli ön işlemeler gerçekleştirilmiş ve hedef değişkenimiz gerekli matematiksel işlemler ile özellik mühendisliği kullanılarak hesaplanmıştır.

**6.4.) İtem\_Data Tablosu:**

Ürünlerin özelliklerinin bulunduğu bu yapıda markalarımız temsil eden brand değişkeni sayısal bir veri olarak sıralı bir şekilde verilmiş bulunmaktadır. Bu aşamada brand üzerinde hiçbir değişiklik yapılmayacaktır. Brand\_type ve category isimli değişkenlerimiz ise Label Encoding yöntemi ile sayısallaştırılacaktır.

|  |
| --- |
| urunLE=LabelEncoder()  urun\_verileri.brand\_type = urunLE.fit\_transform(urun\_verileri["brand\_type"])  urun\_verileri.category = urunLE.fit\_transform(urun\_verileri["category"]) |

**7. Veri Setlerinin Birleştirilmesi Ve Değişken Seçimi**

Gerekli ön işlemeleri yaptıktan sonra regresyon aşamamızda gerekli veri setleri birleştirilerek bir ve özellik seçimi yapılacaktır. Yapılan incelemeler sonucu costumer\_demograpich isimli veri setinde 760 adet müşteri bulunurken costumer\_transaction tablosunda ise bu sayı 1582 adet olduğu tespit edilmiştir. Yani demografik özelliklerini bilmediğimiz 822 adet müşterimiz mevcuttur. Veri seti birleştirme işlemini bu iki tablo arasında yaparken customer\_id değişkenini kullanarak birleştirmemiz gerekmektedir. birleştirme yöntemlerinden inner join methodu kullanıldığı zaman birleştirme sonucu sadece iki veri setindeki ortak olan müşteriler alınacaktır. Demografik özelliklerini bilmediğimiz müşterilerimizi regresyon aşamasında boş hücreler içeren yeni veriler oluşturacağından kullanılmaması gerekmektedir. Birleştirme işlemi aşağıdaki gözüktüğü gibi gerçekleştirilmiştir. Bir sonraki aşamada veriler isimli yeni oluşturulan veri seti ile ürün özellikleri tablosu item\_id ortak anahtar üzerinden birleştirilmiştir.

|  |
| --- |
| veriler=pd.merge(musteri\_islemleri,musteri\_demografik,on='customer\_id',how='inner')  veriler=pd.merge(veriler,urun\_verileri,on='item\_id',how='inner') |

Yapılan bu birleştirme işlemleri sonucu 1.324.566 adet var olan değerimiz 791.602 olarak değişmiştir. Campaign\_Data, Coupon\_İtem\_Mapping veri setlerinin birleştirilmeme sebebi bir ürüne birden fazla kampanya tanımlanması ve bir ürün birden fazla kupon kullanılarak alınmasından ötürü o ürüne uygulanan kupon sayısı kadar var olan kaydın kendisini tekrar etmesi gözlemlenecektir. Bu sebepten ötürü bu veri setleri birleşmemeleri gerekmektedir. Verilerimiz arası gerek ön işlemeden gerekse öznitelik mühendisliğinden kaynaklı olarak kukla değişken (dummy verable) tuzakları oluşmuş durumdadır. Bu sebepten ötürü değişkenlerimizi seçmemiz gerekmektedir.

* **Date :** gün ay ve hafta günü cinsinden değişkenlerimizi bulduğumuzdan ötürü sisteme bu değişkeni sistemden çıkarmamaız gerekecetktir.
* **Customer\_İd :**Müşterilerimizin demografik özelliklerinden veya ürün özelliklerinden tahmin istediğimizden ötürü sistemden çıkarmamız gerekmetkedir. Ayrıca sisteme müşteri id vermemiz ezberlememize sebebiyet verebilir.

**item\_İd :**Müşterilerimizin demografik özelliklerinden veya ürün özelliklerinden tahmin istediğimizden ötürü sistemden çıkarmamız gerekmetkedir. Ayrıca sisteme item\_id vermemiz ezberlememize sebebiyet verebilir.

* **Other\_Discount ve Coupone Discount Değişkenleri:**toplam indirim sistemde zaten mevcut ve bu bizim target hedefimizdir. Bu sebepten ötürü hiçbir şekilde bunları sisteme sokmamamız gerekir. Aksi taktirde dummy verable oluşturacaktır.
* **Selling\_Price:**fiyatları indirimsiz hali üzerinden indirim oranı tahmini yaplıması istendiği için indirimli fiyat değişkeni sistemden çıkartılması gerekmektedir.
* **Toplam\_İndirim :**Toplan indirim yüzdelik cinsinden kaynaklı olarak target değişkenimiz olarak vermiş bulunmaktayız.
* **Marital\_Status :**Yapılan önişlemeler sonucu bu değişkendeki boş sütunların hepsinin doldurulamaması sebebi ile sisteme verilmemesi gerekmektedir.
* **No\_of\_children :**family size ile kukla değişken oluşturabilmekle birlikte ön işleme aşamasında tüm değişkenlerin doldurulamaması sebebi ile sisteme verilmemektedir.

Değişken seçimleride yapıldığına göre bir sonraki başlıklar altında regresyon sonuçları değerlendirilecektir.

**7. Modelleme Aşaması Ve Elde Edilen Sonuçlar**

**7.1 Simple Linear Regressor (SLR):**

Basit doğrusal regresyon olarak bilinen bu algoritma birçok özelliğe sahip olan bu veri setinde kullanılması için uygun değildir. Fakat en yüksek değere sahip değişkenin seçimi için uygulanmıştır. Yapılan işlemler sonucunda veri seti train ve test olarak bölümlendirilmiştir.

|  |  |
| --- | --- |
| bagımsız\_degisken = veriler[['category']]  bagımlı\_degisken=veriler[['yuzde\_kac\_indirim']]  xtrain,xtest,ytrain,ytest=train\_test\_split(bagımsız\_degisken,bagımlı\_degisken,test\_size=0.33, random\_state=0)  lr=LinearRegression()  lr.fit(xtrain,ytrain)  test=lr.predict(xtest)  print("Rsqure degerimiz ",r2\_score(ytest,test))   |  | | --- | | >>  Rsqure degerimiz 0.00597086177238404 | |

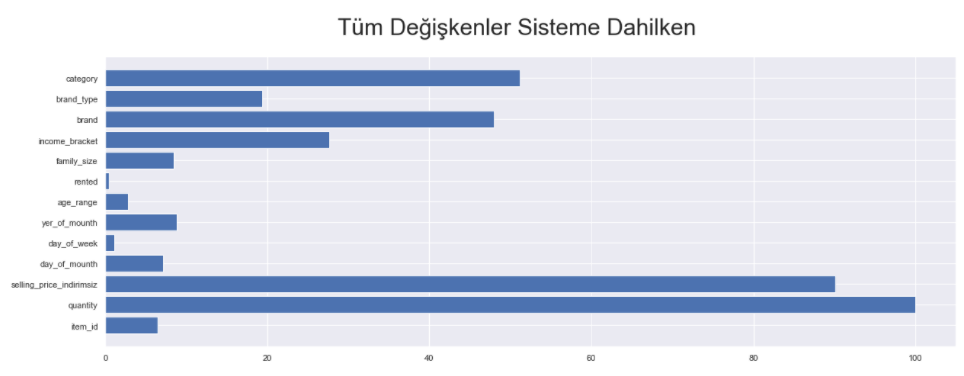
Model Doğrulaması için en bilindik yöntemlerden birisi olan cross validation kullanılmıştır. Yapılan testler sonucu bölümlendirme sayımızı 10 olduğu incelemede en yüksek yakalanan değer 0.015 iken ortalama değerimiz -0.015 olarak hesaplanmıştır. Bir sonraki incelemede veri setinden bir örnek yerine tüm veri setini sisteme vererek bir cross validaton yöntemi ile doğrulama yapılacaktır.

Bir sonraki aşama olan tüm veri setinin eğitilip cross validaton a tabi tutulması sonucu elde edilen değeler -0.015 olarak hesaplanmıştır. Tek bir değişken üzerinden tahminleme yapabilmek kompleks veri setleri için imkânsız sayılmaktadır. Burada gözlemlenmiş bulunulmaktadır.

|  |  |
| --- | --- |
| a=cross\_val\_score(estimator=lr,X=bagımsız\_degisken,y=bagımlı\_degisken,cv=10,scoring="r2")  bolumlendirmek=max(a)  result = np.where(a == bolumlendirmek)  **print**("Veri setini 10 parçaya bölüp yaptığım train ve test sonuçlarında r değerimiz : ",bolumlendirmek, " olarak hesaplanması ile birlikte ortalama : ",a.mean()," . olarak hesaplanmıştır")   |  | | --- | | >>> Veri setini 10 parçaya bölüp yaptığım train ve test sonuçlarında r değerimiz : 0.015982621708820255 olarak hesaplanması ile birlikte ortalama : -0.015823771852639133 | |

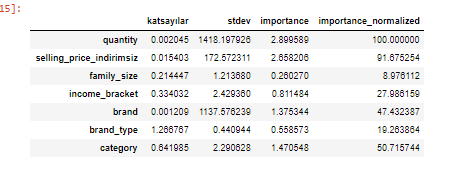
**7.2 Multiple Linear Regressor (MLR):**

Çoklu Doğrusal Regresyon olarak bilinen birden fazla bağımsız değişkenden bir bağımlı değişken tahmini yaptığımız doğrusal regresyon algoritmamız için gerekli değişkenlerin seçilmesi ile bir modelleme daha denenecektir. Tüm değişkenlerin üzerinden %70 lik bir oranda bölüntülere ayırarak yapılarak regresyon denemesi yapıldığı zaman 0.0327 lik bir R2 ve 0.0326 lık bir Adj. R2 değeri hesaplanmıştır. Modelin oluşturulma süresi 0.296 saniye olarak hesaplanmıştır. Gerek modelimizde olumsuz etkilene değişkenlerin çıkartılması ile gerekse modelimizi daha iyi optimize edebilmek adına değişken eleme yöntemleri kullanılarak gereksiz değişkenler elenmesi gerekmektedir. Çalışma boyunca Genellikle Feature importance ağırlıklı olmak üzere korelasyon matrisi ve sarmal (wrapped) method olarak bilinen backward- eliminaton yöntemlerine başvurulmuştur.



Şekil 15 Feature İmportance kullanılarak tüm değişkenlerin oluşturduğu özellik öneminin Grafiği

Feature importance kavramını açmamız gerekirse değişkenlerin modellerimizi ne kadar bir değerde etkilediğinin hesaplanması manasına gelmektedir. Veri setini eğitim ve test ile ayırılması sonucu ve adım adım ilerlenerek sistemi en az etkileyen değişkenler çıkarılıp her seferinde R2 ve modelin oluşturma zamanı hesaplanarak kabul edilebilir en optimum model üretilmeye çalışılmıştır. İşlemler sonucunda son aşamada hem süre bazında hem de regresyon değeri bazında en optimum değişkenler ve bu değişkenlerin özellikleri aşağıda gözüktüğü gibidir.

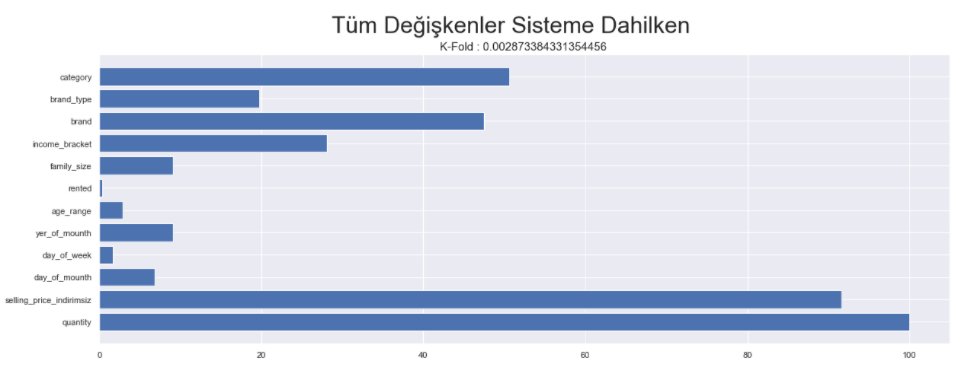


Şekil 16 Feature İmportance yöntemi ile değişken eleme sonucu elde edilen değişkenlerimiz

Elde edilen değişkenlerle birlikte model oluşturma süremiz 0.421 saniye sürmesiyle birlikte sürmüştür. Bu aşamada R2 değerimiz ise 0.0322 olarak son halini almıştır. Oluşturulan modelin Feature importance değerleri başta olmak üzere katsayıları ve standart sapmaları şekil 16’da gösterilmiştir. Oluşturulan model Cross validation ile CV=10 parametresi için hesaplandığı zaman rsquare değeri 0.0025 olarak hesaplanmıştır. Anlaşılacağı üzere model sadece train ve test verisi için geçerli olup diğer değişkenlerimiz için doğru bir hesaplama gerçekleştirememektedir. Bu aşamada tüm veri setini alarak ve her aşamada cross validaton yapılarak model doğrulamaya çalışılacaktır.

Backward eliminaton yönteminde elde edilen başarım feature importance değerinden daha düşük ve daha fazla değişken üzerinden tahmin edildiği için yapılan hesaplamalar neticesinde feature importance ile işlemlere devam edilmiştir. Backward eliminaton ile 0.028 R2 elde edilmiştir.

Veri setindeki tüm verilerin uydurulabilmesi için bu aşamada tüm veriler sisteme dahil edilerek bir hesaplama yapılmıştır. Yapılan incelemeler neticesinde tüm değişkenlerin sisteme dahil olduğu zaman model hesaplama süresi 0.484 saniye olarak hesaplanmıştır.



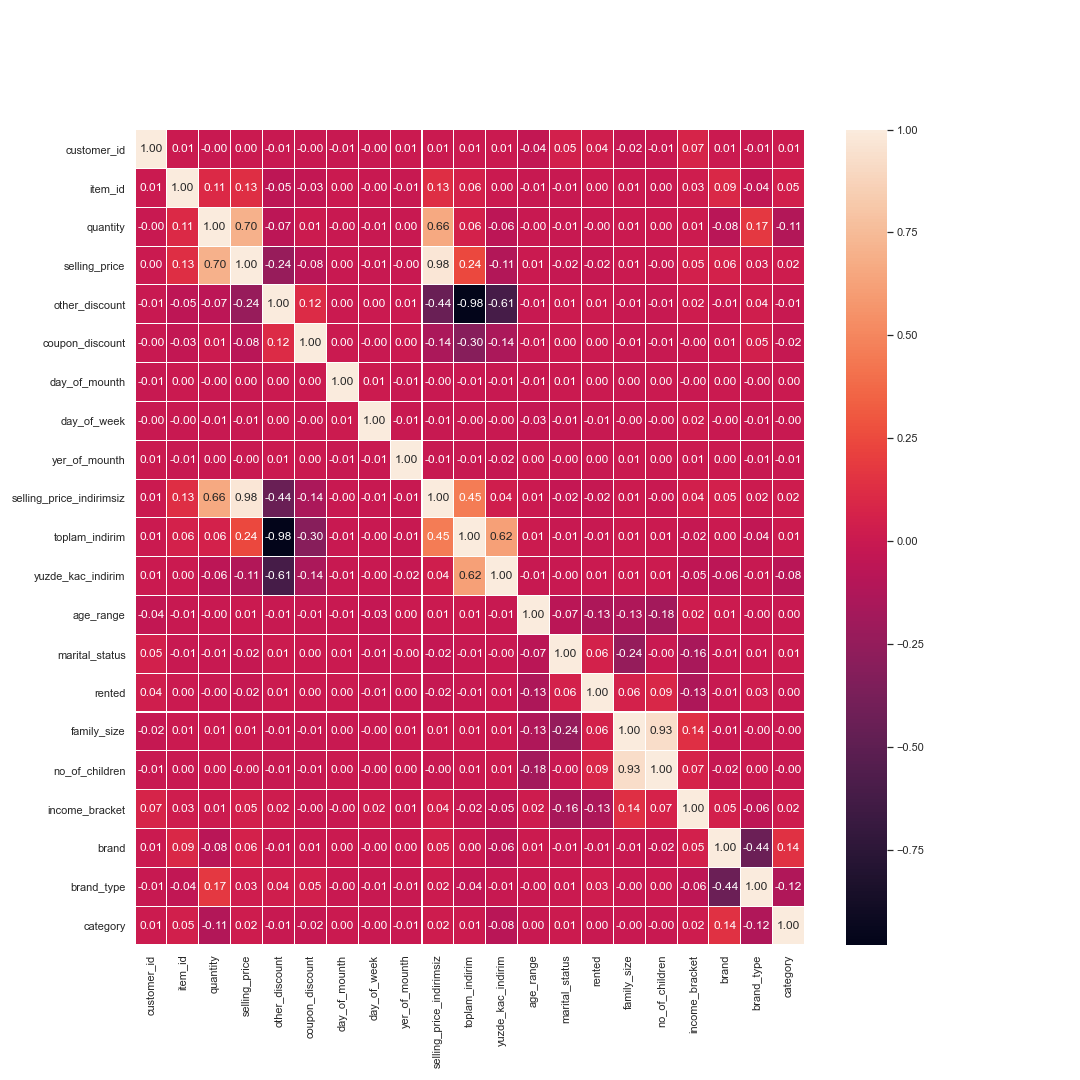
Şekil 17 Tüm değişkenlerin sisteme dahil olduğu zamanki feature importance değerleri

Yapılan son aşamadaki değişken elemelerle birlikte elde edilen değeler R2 olarak0.0027 olarak hesaplanmış bulunmaktadır ve 0.28 saniye sürmüştür. Veri seti doğrusal bir modellemeye uygun olmadığının kanıtlanması yapılmış olup bir sonraki aşamada doğrusal olmayan modellerin hesaplamasında kullanılan algoritmalar incelenerek ilerlenecektir.



Şekil 18 MLR son hali ve değerleri

Değerlerimiz arasındaki ilişkiyi ısı haritası olarak aktarmak gerekirse aşağıdaki şekilde ısı haritamız mevcuttur.

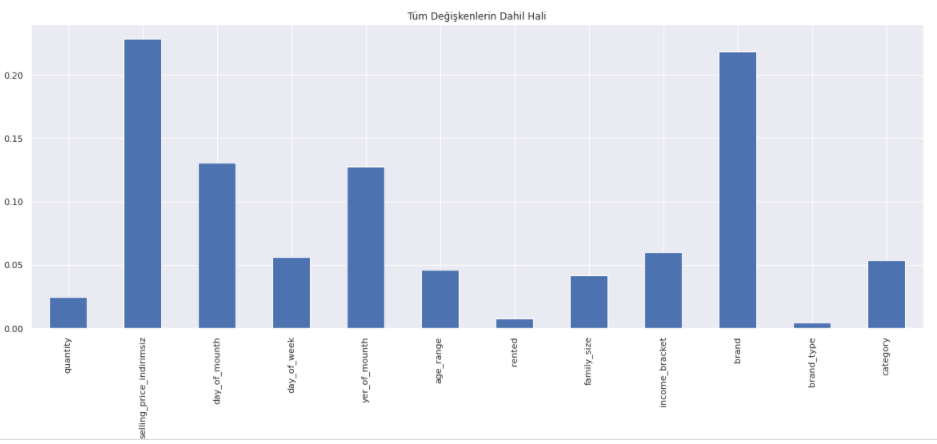


Şekil 19 Isı Haritası

**7.3 Karar Ağaçları (DT):**

Temelinde Sınıflandırma Problemlerini baz alan bu algoritmamız doğrusal olmayan regresyon tiplerinde sıklıkla kullanılmaktadır. Verilerin entropilerini baz alarak bir ağaç yapısını oluşturan bu algoritmamız çalışma süresince kullanılmıştır. Verilerimizin %70 ini eğitim ve geri kalanını test kümesi olarak ayırmamızın ardından sklearn.tree kütüphanesinin altından DesicionTreeRegressor olarak çağırılmıştır.

|  |  |
| --- | --- |
| bagımsız\_degisken = veriler[[ 'quantity', 'selling\_price\_indirimsiz', 'day\_of\_mounth', 'day\_of\_week', yer\_of\_mounth', 'age\_range', 'rented', 'family\_size','income\_bracket', 'brand', 'brand\_type', 'category'  ]]  bagımlı\_degisken=veriler[['yuzde\_kac\_indirim']]  xtrain,xtest,ytrain,ytest=train\_test\_split(bagımsız\_degisken,bagımlı\_degisken,test\_size=**0.33**,random\_state=**0**)  start = time.process\_time()# Model çalışma süresi hesaplanması istendiği için sayaç başlatılmıştır.  regressor = DecisionTreeRegressor(random\_state=**0**)  regressor.fit(xtrain,ytrain)  tahmin= regressor.predict(xtest)  end = time.process\_time()# işlem bu aşamada bitmiş bulunmaktadır.  r2=r2\_score(ytest,tahmin)  r2  **print**("Rsqure degerimiz ",r2,"ve Adj. R^2 değerimiz :" ,stats.adj\_r2\_score(regressor, xtest,ytest) )  **print** ("model boyunca geçen süremiz : ",end - start ," saniye sürdü")   |  | | --- | | >>>  Rsqure degerimiz 0.20634221681115628 ve Adj. R^2 değerimiz : 0.20630575697179632 model boyunca geçen süremiz : 4.1713579150000015 saniye sürdü | |

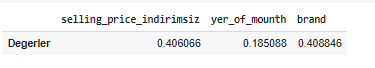


Şekil 20 Karar Ağacı Modelinin Tüm değişkenlerin sisteme verilmesi durumundaki feature importance değerleri

Karar ağacı algoritmasının feature importance değerlerini hesaplamamız için lineer regresyon modellerindeki gibi birtakım sayısal işlemlerden geçmesine gerek yoktur. Oluşturulan ve eğitilmiş modelin ismiyle birlikte feature\_importances\_ isimli fonksiyonunu çağırmamız yeterlidir. Karar ağaçları bu feature importance değerlerini kendileri hesaplayabilmektedirler.

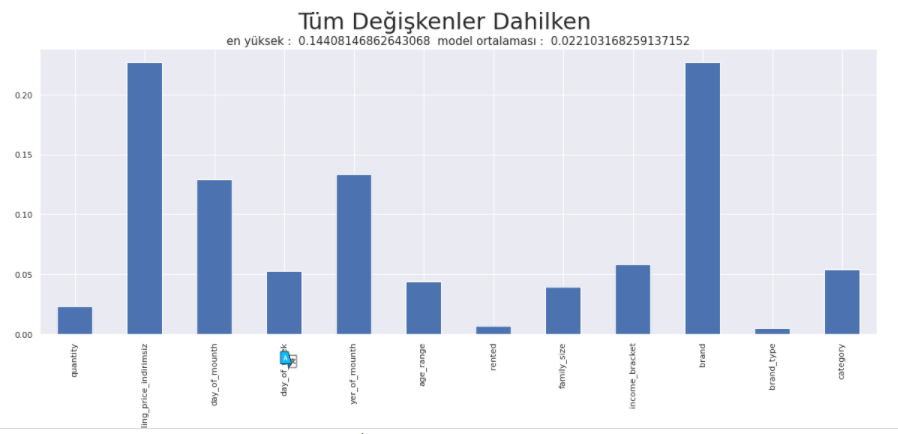
Yapılan analizler sonucu tüm değişkenlerimiz sisteme dahilken R2 değerimiz 0.206 olarak hesaplanmakta ve model eğitim süresi 4.17 saniye sürmektedir. Daha öncesinde doğrusal regresyon modelinde denenmiş olan korelasyon matrisi ve backward-eliminaton yöntemlerinin belirsizliği sebebi ile bu ve sonraki modellerde değişken elemekte feature importance kullanılacaktır.

Yapılan elemelerin ardından en verimli değişkenler tespit edilmiştir. Bu değişkenlerimiz sırası ile year\_of\_mounth, selling\_price\_indirimsiz ve brand değişkenleridir. Bu seçim işlemi sonucu R2 değerimiz 0.449’a yükselmekle birlikte model oluşturulma süresi 1.67 saniye olarak sürmektedir. Değerlere bakıldığında tahmin değerimizde %117 lik bir artış ve eğitim sürecinden ise %181 lik bir kazanç sağlanmıştır.



Şekil 21 Seçilen değişkenlerin feature importance değerleri

Üretilen yeni modelin doğrulaması için cross-validaton sürecinden geçmesi üzerine en yüksek korelasyon değerimiz 0.296 model ortalaması ise 0.181 olarak bulunmuştur. Anlaşılan bu seçim işlemleri sebebi ile modelimiz ezberleme sürecine girmiş bulunmaktadır. Bu sebepten tüm veri setini modelimize dahil ederek doğruluk değeri olarak cross validaton belirlenmesi üzerine çalışmalar gerçekleştirilmiştir.



Şekil 22 Tüm veri setinin dahil edilmesi sonucu oluşan yeni feature importance değerleri

Çalışma süreci boyunca düzenli olarak kullanılan feature importance değerlerine göre seçim yapma işlemi bu aşamada da devam etmiş bulunmaktadır. Yapılan seçimler sonucu en verimli değişkenlerimiz sırası ile selling\_price\_indirimsiz, year\_of\_mounth, ve brand olarak belirlenmiştir. Bu değişkenler ile cross-validaton dan dönen değerler karşılaştırıldığında en yüksek korelasyon değerimiz : 0.379 model ortalaması ise 0.247 olarak tespit edilmiştir.

Elde edilen modelin hiperparametre optimizasyonu için genellikle karar ağaçları için en önemli ve regresyon sonucunda büyük etkisi olan parametreler seçilerek ve GridSearch yöntemi kullanılarak hiperparametre optimizasyonu gerçekleştirilmiştir.

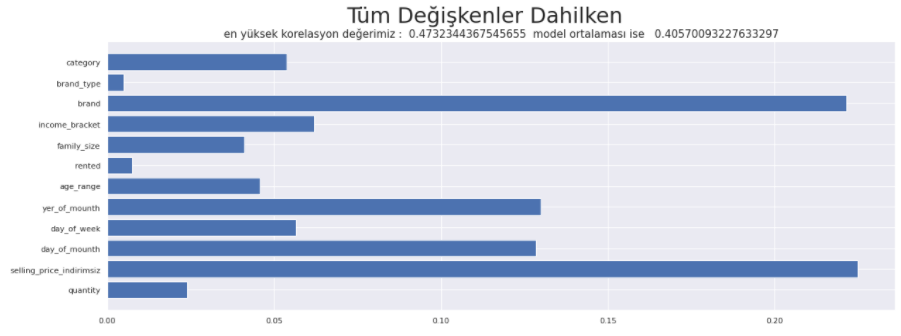
|  |  |
| --- | --- |
| bagımsız\_degisken = veriler[[ 'selling\_price\_indirimsiz','yer\_of\_mounth', 'brand',]]  bagımlı\_degisken=veriler[['yuzde\_kac\_indirim']]  regressor = DecisionTreeRegressor(random\_state=0)  parametreler = {  'max\_depth':range(1, 100),  'splitter': ['best',"random"],  "min\_samples\_split":[2,3,4,5,6,7,8],  }  GR=GridSearchCV(estimator=regressor,param\_grid=parametreler,scoring="r2",verbose=5)  sonuclar=GR.fit(bagımsız\_degisken,bagımlı\_degisken)  deger=sonuclar.best\_params\_  deger   |  | | --- | | >>>  {'max\_depth': 24, 'min\_samples\_split': 8, 'splitter': 'best'} | |

Hiperparametre optimizasyonu ile birlikte en yüksek korelasyon değerimiz : 0.379 model ortalaması ise 0.280 olarak hesaplanmıştır. Eğtim süresi 1.68 saniye sürmüştür. Başlangıçta tüm değişkenler sisteme dahilken 5.53 saniye sürmüştür.

**7.4 Rassal Orman Regresyonu (RFR):**

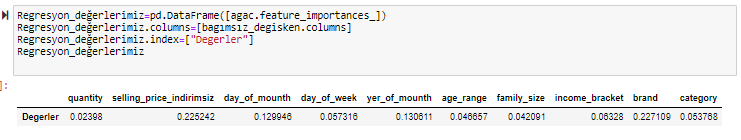
Ensemble learning altında bulunan rassal orman regresyonu modeli kullanılarak bir modelleme daha geliştirilmiş bulunmaktadır. Birden fazla karar ağacına bağlı olarak yorum yapan algoritma genellikle normal bir karar ağacından daha başarılı tahminler yapılabilmektedir. Çalışmanın gerçekleşebilmesi için daha önceki algoritmalarımızda %70 ve %30 oranlarında veri setlerimizi bölerek eğitim ve tahmin aşamalarını gerçekleştirmemiz sonucu veri setinde bir ezberleme ve seçilen veri setinin diğer veri setleri üzerinde uyumsuzluğu ve kabul edilemeyecek bir doğrulama sayısına sahip olmasından ötürü bu aşamada tüm veri seti dahil edilerek bir eğitim ve değişken eleme aşamaları gerçekleşmiştir.

Başlangıçta daha hızlı tahminler yapabilmesi için model oluşturmada ağaç sayısını 50 vererek hesaplamalara başladığımız zaman modelin oluşturma süresi tam olarak 183.13 saniye sürmekte ve corss validaton değerimiz 5 seçilerek yapılan testler sonucu en yüksek değer 0.473 iken çıkan değerler ortalaması 0.405 olarak hesaplammıştır.



Şekil 23 Rassal orman özellik önemleri

Yapılan değişken eleme yöntemleri neticesinde en son aşamada rented değişkeninin çıkartılmasıyla birlikte en optimize halini almış bulunmaktadır. Özellik önemleri ve değişkenler bir alt sayfada belirtilmiş bulunamaktadır. Elde edilen değerler neticesinde en yüksek korelasyon değerimiz 0.472, ortalama değerimiz ise 0.405 olarak hesaplanmıştır modelin oluşturulma süresi 191 saniye sürmektedir.



Şekil 24 Değişkenlerimizin Feature İmportance değerlerinin tablo şeklinde gösterilmesi

Değerlerin iyileşmesi ile ve beraberinde tahmindeki doğrulukları artması sebebi ile kullanıma uygun bir model olduğunun tespiti gerçekleştirilmiş ve modelimizin hiperparametre optimizasyonlarının yapılması gerekmektedir.

|  |
| --- |
| **from** **sklearn.model\_selection** **import** GridSearchCV  random\_grid = {'n\_estimators':range( 100, 400),  "criterion":["mse"],  }  agac=RandomForestRegressor(random\_state=0)  GS = GridSearchCV(estimator=agac,param\_grid= random\_grid,verbose=1,n\_jobs=-1)  GS.fit(bagımsız\_degisken, bagımlı\_degisken) |

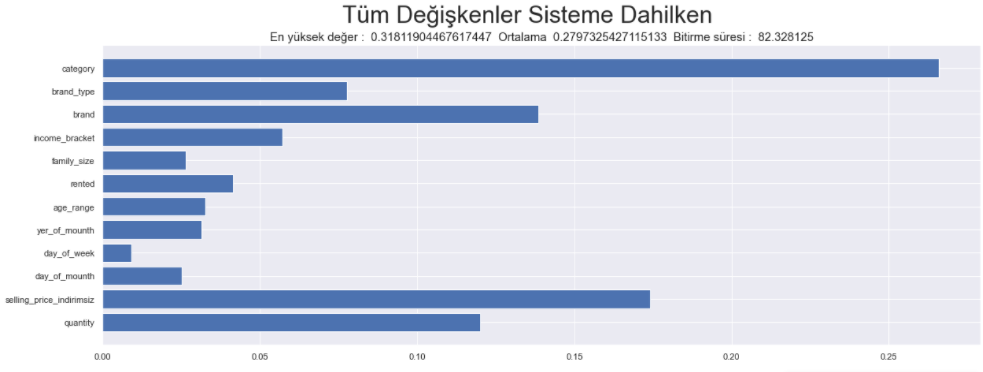
Rassal ormanda birden çok parametre verilebilmektedir. Verilebilecek parametreler arasından en tutarlı olanı n\_estimators olarak bilinmektedir. Tahminimizi arttıracak etkenlerden birisi ise ağaç sayısı olarak adlandırabileceğimiz n\_estimators sayısıdır. Bu aşamada daha fazla ağaç verilebilirdi fakat sistem yetersizliğinden kaynaklı olarak bu sayının daha uygun olabileceği düşünüldü. Yapılan hiperparametre optimizasyonunda bu değer 300 olarak bulundu ve yeni kurulan model için hesaplamamız sonucu olarak çıktı aşağıda verilmiştir.

|  |
| --- |
| >>>  en yüksek değer : 0.4808290343669799 Ortalama 0.41467808666731587 |

**7.5 XGBOOST (XGBOOST):**

Extreme gradient boostingalgoritması sektörel bazda en bilindiklerinden olan ve performansı ve başarısı ile kullanım için çokça tercih edilen algoritmalardan birisidir. Yüksek verilerde iyi performans gösterebilmesi, hızlı çalışması, hafızayı verimli kullanması ve modelin ve problemin yorumunu mümkün kılan bir algoritmadır. Birçok platformda mevcut olan bir algoritmadır.

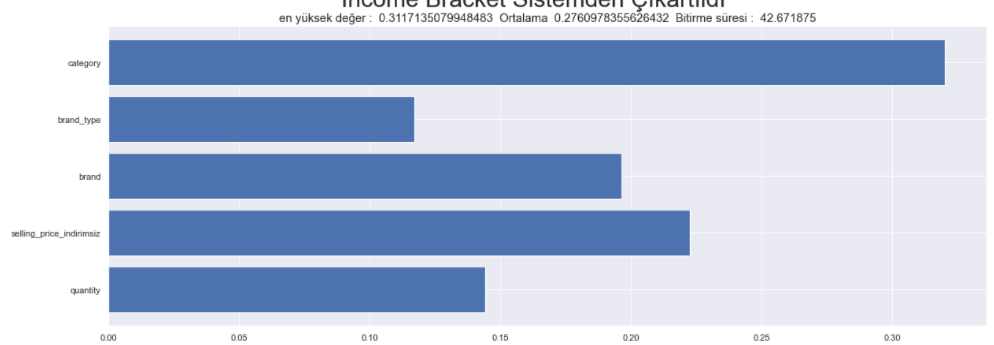
Daha önceki uygulamalarda olduğu gibi train ve test olarak ayırmamız veri setinde ezberlemeye sebebiyet verebilme ihtimaline karşılık olarak tüm veri seti üzerinden bir öğrenim gerçekleştirilecektir.



Şekil 25 XGBOOST algoritması tüm veri seti dahilken feature importance değerleri

|  |  |
| --- | --- |
| bagımsız\_degisken = veriler[[ 'quantity', 'selling\_price\_indirimsiz', 'day\_of\_mounth', 'day\_of\_week','yer\_of\_mounth','age\_range',  'rented', 'family\_size','income\_bracket', 'brand', 'brand\_type', 'category'  ]]    bagımlı\_degisken=veriler[['yuzde\_kac\_indirim']]  start = time.process\_time()  xgb\_model = xg.XGBRegressor(random\_state=0)  xgb\_model.fit(bagımsız\_degisken,bagımlı\_degisken)  end = time.process\_time()  agac=xg.XGBRegressor(random\_state=0)  a=cross\_val\_score(estimator=agac,X=bagımsız\_degisken,y=bagımlı\_degisken,cv=5)  bolumlendirmek=max(a)  result = np.where(a == bolumlendirmek)  **print**("en yüksek değer : ",bolumlendirmek, " Ortalama " ,a.mean()," Bitirme süresi : ",end-start)   |  | | --- | | >>>  en yüksek değer : 0.31811904467617447 Ortalama 0.2797325427115133 Bitirme süresi : 82.328125 | |

Yapılan bu eğitim sonucunda feature importance yöntemi kullanılarak değişken eleme başlatılmış olup yeterli değerlini sağlayamayan hem de modelimizi yavaşlatmaya sebebiyet veren değişkenlerin sırası ile elenmesi sonucu sistemde 5 değişken kalarak eğitim süremizi 42 saniyede tamamlayabilmesiyle birlikte R2 değerimiz 0.276 olarak bulunmuştur. Regresyon değerimizden önemsenmeyecek kadar düşük bir kayıp verilmesi ile birlikte beraberinde süreden yarı yarıya bir kazanç sağlanmıştır.



Şekil 15 Parametreler belirlendi

Model parametrelerinin belirlenmesiyle birlikte modelimiz daha az değişkenden daha doğru tahminler yapabilir hale ve zaman ile performanstan kazanç elde edilebilir gelmiş bulunmaktadır. Bu aşamadan sonra yapılması gereken ise modelimizin bu değişkenler altındaki hiperparametre optimizasyonunu gerçekleştirmektir. Hiperparametre optimizasyonunu için XGBoost algoritmasında birçok parametre bulunmaktadır. Fakat bu algoritmanın en etkili parametreleri ise öğrenme oranı ve ağaç sayısı değişkenleridir. Bu sebepten ötürü bu iki değişken üzerinde farklı aralıklar kullanılarak bir GridSearch araması gerçekleştirilerek en etkili parametreleri bulmaya çalışacağız

|  |
| --- |
| parameters={  "learning\_rate":[0.1,0.01,0.001],  "n\_estimators":range(100, 1200, 100),  }  xgb\_model = xg.XGBRegressor(random\_state=0)  GS=GridSearchCV(estimator=xgb\_model,param\_grid=parameters, verbose=10,n\_jobs=-1) |

K-fold sayımızı varsayılan parametremizi vermenin ardından toplamda 165 adet eğitim gerçekleşmiş bulunaktadır. Bu eğtimler sonucunda öğrenme oranımız 0.1 ve ağaç sayımız 1100 olarak elde edilmiştir. Sonuç olarak

|  |  |
| --- | --- |
| bagımsız\_degisken = veriler[[ 'quantity', 'selling\_price\_indirimsiz', 'brand', 'brand\_type', 'category'  ]]    bagımlı\_degisken=veriler[['yuzde\_kac\_indirim']]  optimizasyon=xg.XGBRegressor(learning\_rate=0.1,n\_estimators=1100)  a=cross\_val\_score(estimator=optimizasyon,X=bagımsız\_degisken,y=bagımlı\_degisken,cv=5)  bolumlendirmek=max(a)  result = np.where(a == bolumlendirmek)  **print**(a.mean()," K-Fold Değerimizin ortalaması olamkla birlikte en yüksek k fold değeri ", a.max())   |  | | --- | | **>>>**  0.3176707760387738 K-Fold Değerimizin ortalaması olmakla birlikte en yüksek k fold değeri 0.3636535916250939 | |

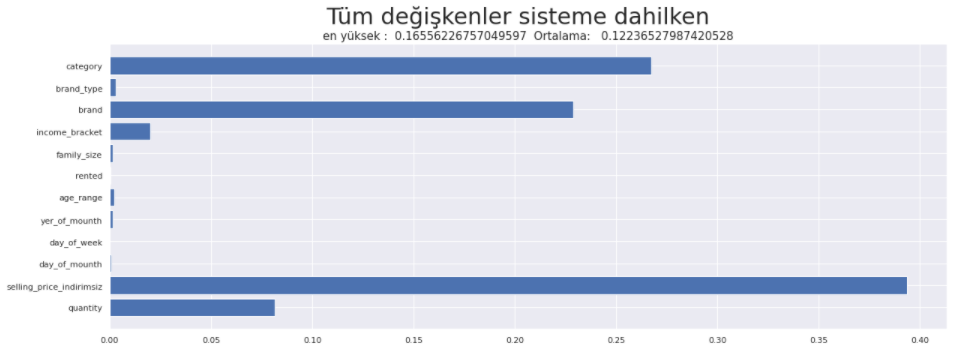
Yapılan son optimizasyonların beraberinde R2 değerimiz 0.317 olarak hesaplanmış bulunmaktadır.

**7.6 Gradient Boosting Regression:**

AdaBoost’un sınıflandırma ve regresyon problemlerine kolayca uyarlanabilen genelleştirilmiş versiyonudur. Bir boosting algoritması olması sebebi ile genellikle başarılı sonuçlar gözlemlenebilmektedir.

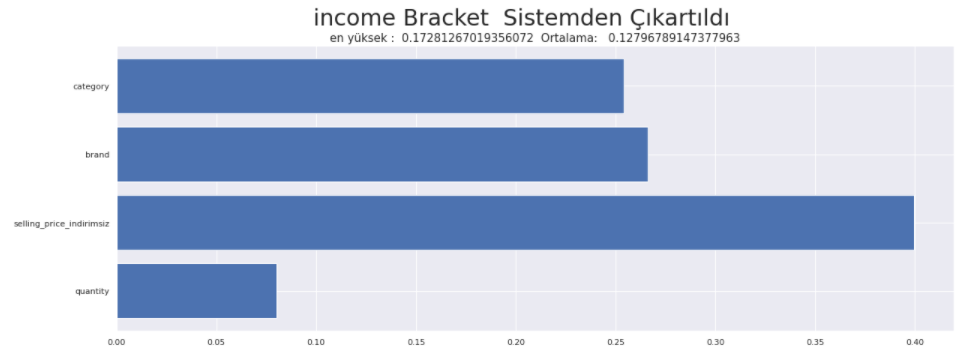
Algoritma diğer modellerimizde olduğu gibi ezberlemeye gitmemesi için tüm veri seti dahil edilerek işe başlanmıştır.

|  |  |
| --- | --- |
| bagımsız\_degisken = veriler[[ 'quantity', 'selling\_price\_indirimsiz', 'day\_of\_mounth', 'day\_of\_week','yer\_of\_mounth', 'age\_range', 'rented', 'family\_size','income\_bracket', 'brand', 'brand\_type', 'category'  ]]  bagımlı\_degisken=veriler[['yuzde\_kac\_indirim']]  start = time.process\_time()  xgb\_model = GradientBoostingRegressor(random\_state=0)  xgb\_model.fit(bagımsız\_degisken,bagımlı\_degisken)  end = time.process\_time()  GBR=GradientBoostingRegressor(random\_state=0)  a=cross\_val\_score(estimator=GBR,X=bagımsız\_degisken,y=bagımlı\_degisken,cv=5)  bolumlendirmek=max(a)  result = np.where(a == bolumlendirmek)  **print**("en yüksek : ",bolumlendirmek, " Ortalama: " ,a.mean()," süre: ",end-start)   |  | | --- | | >>>  en yüksek : 0.16556226757049597 Ortalama: 0.12236527987420528 süre: 103.859375 | |



Şekil 16 Tüm değişkenlerin dahil olması ile kurulan modelin feature importance değerleri

Yapılan işlemler sonucu 103 saniye kadar sürerek kurulmuş olan ve şekil 16 da gösterildiği gibi feature importance değerlerine sahip olan modelimizde birçok değişken şekilde gözüktüğü üzere sisteme herhangi bir etkileri bulunamamaktadır. Beraberinde bu değişkenimizde hem doğru hem hata payımızı düşürmek hem de modellerin daha hızlı oluşturulabilmesi için bu aşamada değişkenlerimizi feature importance değerlerine bakarak eleme işlemleri yapılması gerekmektedir. Yapılan elemeler sonucu en optimum değerler bulunmuş olup değerleri ve değişkenleri aşağıdaki şekilde mevcuttur.



Şekil 17 GBR için değişken seçimi sonucu elde edilen değerler

Bu elemelerin sonucunda değerlerimiz kabul edilebilir seviyenin altında kalmaktadır. Değerleri yükseltebilmek ve daha kullanılabilir bir model elde edebilmek adına bu aşamadan hiperparametre optimizasyonu gerçekleştirmemiz gerekmektedir. son aşamada eğitim süremiz 40 saniyeye düşmüştür. Gradient Boosting Regression algotirması için birçok hiperparametre bulunmakla birlikte sistemi en çok etkileyecek iki parametre üzerinde hiperparametre optimizasyonu gerçekleştirilmiştir. Ağaç sayısı ve öğrenme oranları bu hiperparametrelerdir.

|  |  |
| --- | --- |
| parameters={  "learning\_rate":[0.1,0.01,0.001],  "n\_estimators":range(100,1200,100),  }  gbr\_model = GradientBoostingRegressor(random\_state=0)  GS=GridSearchCV(estimator=gbr\_model,param\_grid=parameters, verbose=10,n\_jobs=-1)  GBR=GS.fit(bagımsız\_degisken,bagımlı\_degisken)  GBR.best\_params\_   |  | | --- | | >>>  {'learning\_rate': 0.1, 'n\_estimators': 1100} | |

Yapılan bu hiperparametre sonucunda en iyi parametreler olarak öğrenme oranı için 0.1 ve ağaç sayısı için 1100 olarak belirlenmiştir. Belirlenen bu parametreler sonucunda son aşamadaki sistem parametreleri bu hiperparametreler ile belirlenmesiyle birlikte yeni oluşturulan model sonucu aşağıdaki kodda mevcuttur.

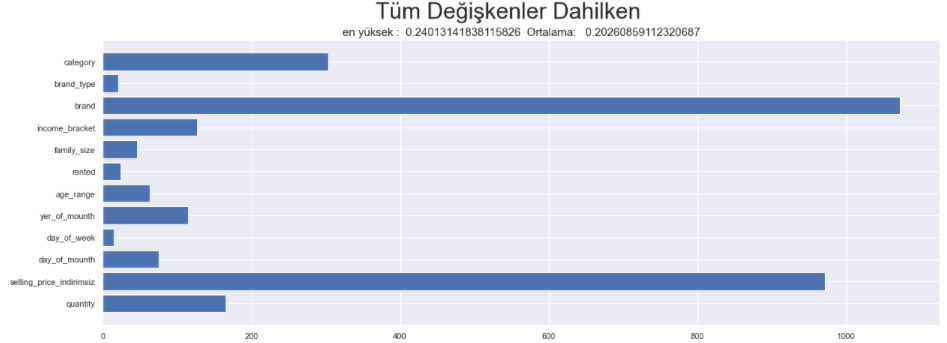
|  |  |
| --- | --- |
| gbr\_model = GradientBoostingRegressor(random\_state=0,learning\_rate=0.1,n\_estimators=1100)  a=cross\_val\_score(estimator=gbr\_model,X=bagımsız\_degisken,y=bagımlı\_degisken,cv=5,n\_jobs=-1)  bolumlendirmek=max(a)  result = np.where(a == bolumlendirmek)  **print**("en yüksek : ",bolumlendirmek, " Ortalama: ",a.mean())   |  | | --- | | >>>  en yüksek : 0.26934405762684543 Ortalama: 0.23208231284967534 | |

**7.7 LightGBM Rgression:**

LightGBM XGBoost’un eğitim süresi kısalması ve performansını arttırmaya yönelik olarak geliştirilmiş olan bir GBM türüdür. 2017 yılında Microsoft tarafından geliştirilmiş olan bu algoritma açık kaynak kodlu olarak paylaşılmıştır. XGboost Regresyonuna nazaran daha hızlı ve performanslı olan bu algoritma Leaf-wise ile bölme işlemlerini gerçekleştirmesi stratejisini izlemektedir. depth-first search yaklaşımını benimsemiştir.

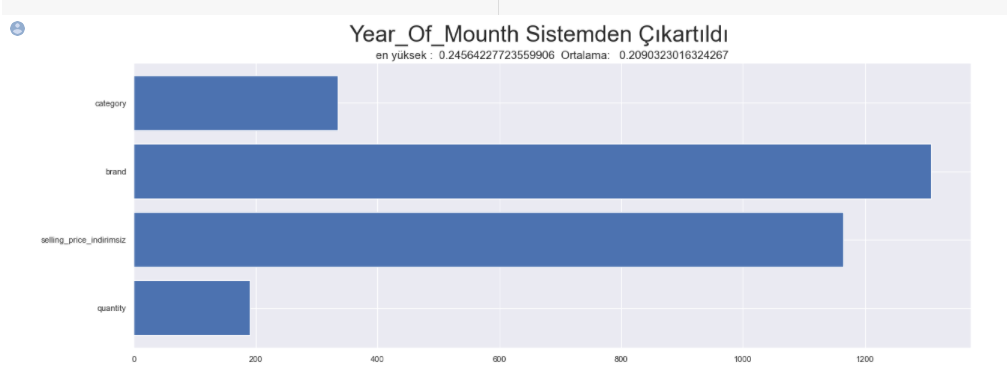
Hız ve performans anlamında XGBoost algoritmasının iyileştirilmiş bir versiyonu olan LightGBM için öncelikle tüm veri setimizi sisteme dahil etmemiz gerekmektedir. Tüm veri setinin sisteme verilerek yapılan yeni regresyon sonucu bir alt şekilde belirtildiği gibidir.

|  |  |
| --- | --- |
| bagımsız\_degisken = veriler[[ 'quantity',  'selling\_price\_indirimsiz', 'day\_of\_mounth', 'day\_of\_week', 'yer\_of\_mounth',  'age\_range', 'rented', 'family\_size', 'income\_bracket', 'brand', 'brand\_type', 'category'  ]]  bagımlı\_degisken=veriler[['yuzde\_kac\_indirim']]  start = time.process\_time()  lgbm = LGBMRegressor(random\_state=0)  lgbm.fit(bagımsız\_degisken,bagımlı\_degisken)  end = time.process\_time()  lgbm2=LGBMRegressor(random\_state=0)  a=cross\_val\_score(estimator=lgbm2,X=bagımsız\_degisken,y=bagımlı\_degisken,cv=5)  bolumlendirmek=max(a)  result = np.where(a == bolumlendirmek)  print("en yüksek : ",bolumlendirmek, " Ortalama:  " ,a.mean()," Süre:  ",start-end )   |  | | --- | | >>>  en yüksek : 0.24013141838115826 Ortalama: 0.20260859112320687 Süre: 7.64 | |



Şekil 18 Tüm değişkenlerin dahil olma durumundaki LightGBM için feature importance denemesi

Yapılan analizler çerçevesinde değişkenlerimizin elediğimizde en optimum değişkenleri bulmamız sonucu ortaya çıkan değerler aşağıdaki şekil üzerinde gösterilmektedir. Tüm değişkenlerin dahil olma durumunda 7.64 saniyede eğitim süresini tamamlarken bu aşamada 5.343 saniyeye kadar bir düşüş gerçekleşmiştir.



Şekil 19 LGBM algoritması için feature importance değerlerine bakarak yapılan elemeler sonucunda elde edilen değişkenler ve regresyon sonuçları

|  |
| --- |
| bagımsız\_degisken = veriler[[ 'quantity', 'selling\_price\_indirimsiz', 'brand', 'category' ]]  bagımlı\_degisken=veriler[['yuzde\_kac\_indirim']]  start = time.process\_time()  lgbm = LGBMRegressor(random\_state=0)  lgbm.fit(bagımsız\_degisken,bagımlı\_degisken)  end = time.process\_time()  lgbm2=LGBMRegressor(random\_state=0)  a=cross\_val\_score(estimator=lgbm2,X=bagımsız\_degisken,y=bagımlı\_degisken,cv=5)  bolumlendirmek=max(a)  result = np.where(a == bolumlendirmek)  print("en yüksek : ",bolumlendirmek, " Ortalama:  " ,a.mean()," Süre:  ",end-start)  en yüksek : 0.24564227723559906 Ortalama: 0.2090323016324267 Süre : 5.34375 |

Yapılan işlemler neticesinde model başarısını arttırabilmesi için yapılan hiperparametre optimizasyonları gerçekleştirilecektir. Belirlenen hiperparametreler bir alt kısımda gözüktüğü gibidir.

|  |  |
| --- | --- |
| parameters={  "learning\_rate":[0.1,0.5,0.01,0.05,0.001,0.005],  "n\_estimators":range(100,1200,100),  }  lgbm = LGBMRegressor(random\_state=0)    GS=GridSearchCV(estimator=lgbm,param\_grid=parameters, verbose=10,n\_jobs=-1)  GBR=GS.fit(bagımsız\_degisken,bagımlı\_degisken)   |  | | --- | | >>> {'learning\_rate': 0.5, 'n\_estimators': 600} | |

Bulunan en iyi parametreler ile modelleme gerçekleştirilmesi sonucu elde edilen değerler en yüksekte 0.352 ve ortalama olarak 0.299 verilmiştir.

|  |  |
| --- | --- |
| lgbm2=LGBMRegressor(random\_state=0,learning\_rate=0.5,n\_estimators=600)  a=cross\_val\_score(estimator=lgbm2,X=bagımsız\_degisken,y=bagımlı\_degisken,cv=5)  bolumlendirmek=max(a)  result = np.where(a == bolumlendirmek)  **print**("en yüksek : ",bolumlendirmek, " Ortalama: " ,a.mean())   |  | | --- | | >>>  en yüksek : 0.3522005997834553 Ortalama: 0.2986659419240375 | |

Performans ve hız bakımından gayet tatminkâr değerler elde edilmiştir. Beraberinde karar ağaçlarından sonra en hızlı çalışan algoritma olarak belirlenmiş ve doğruluk değeri yeterli bulunmuştur.

**8.) Modellerin Değerlendirilmeleri Ve Başarı Analizleri:**

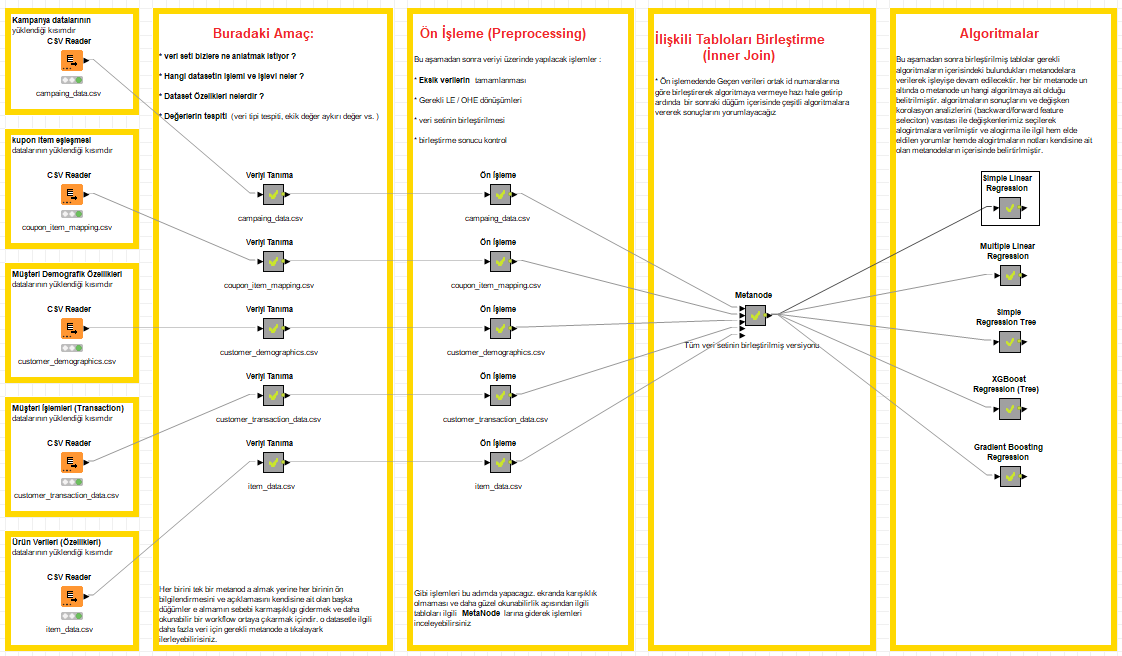
Bu bölüm kısa bir şekilde yapılan çalışmalar hakkında özet geçilmiştir. Verilerimiz doğrusal olarak regresyona tâbi tutulamayan bir veri seti olduğu için doğrusal regresyon tiplerimizin başarısı bu aşamada düşük kalmıştır. Kurulan modellerin hızlılığı ağaç yapılı regresyon algoritmalarına göre daha hızlı gerçekleştirilmektedir. Beraberinde en yüksek değeri orman algoritmasına almış bulunduk. Model kurulum süreleri ve eğitim süreleri her ne kadar uzun olsa dahi eğer doğruluk bizim için ön planda ise rassal orman algoritmalarının kullanılması daha doğru olacaktır. Hem profesyonel iş süreçlerinde hem de yarışmalarda en çok bilindik ve başarısı ile kullanımı sağlayan XGBoost bu aşamada kabul edilebilir değerler sergilemiştir. Hız manasında ise en yüksek değerli veren ve hızlı sayılabilecek olan (XGBoostun yaklaşık onda biri kadar) modelimiz ise lightgbm algoritması ile oluşturulmuştur. XGBoosta göre regresyon değeri düşük sayılsa dahi kabul edilebilir bir değerdedir. Burada en başarılı regresyon değerlerimiz rassal orman algoritması ve en hızlı ve başarılı modelleme için ise lightgbm algoritması seçilmiştir. Değerler bir aşağıdaki tabloda mevcut bulunmaktadır.

|  |  |
| --- | --- |
| **Algoritmalar** | **Başarım Oranları** |
| Simple Linear Regression **(SLR)** | -0.015 |
| Multiple Linear Regression **(MLR)** | 0.0290 |
| Desicion Tree Regression **(DTR)** | 0.279 |
| Random Forest Regression **(RFR)** | 0.414 |
| Extreme Gradient Boosting **(XGBoost)** | 0.317 |
| Gradient Boosting Tree **(GBR)** | 0.232 |
| Light GBM **(LGBM)** | 0.299 |

**9. Knime Üzerinde Çalışmanın Uygulanması**

Knime üzerindeuçtan uca veri bilimi işi yürütebilmemizi sağlayan, açık kaynak kodlu, data mining suit olarak bilinen bir yazılımdır. Çalışma boyunca birçok farklı eklentisi bulunması, farklı platformlar tarafında çalışabilmesi, birçok işletim sistemi ve sunucu sistemi için türevlerinin bulunması, ücretsiz olması avantajlarından ötürü Knime kullanışlı bir yazılımdır. Uygulamalar bu aşamada yerel bir bilgisayarda Knime Analytics Platform’u 4.3.2 sürümü üzerinde gerçekleştirilecektir.

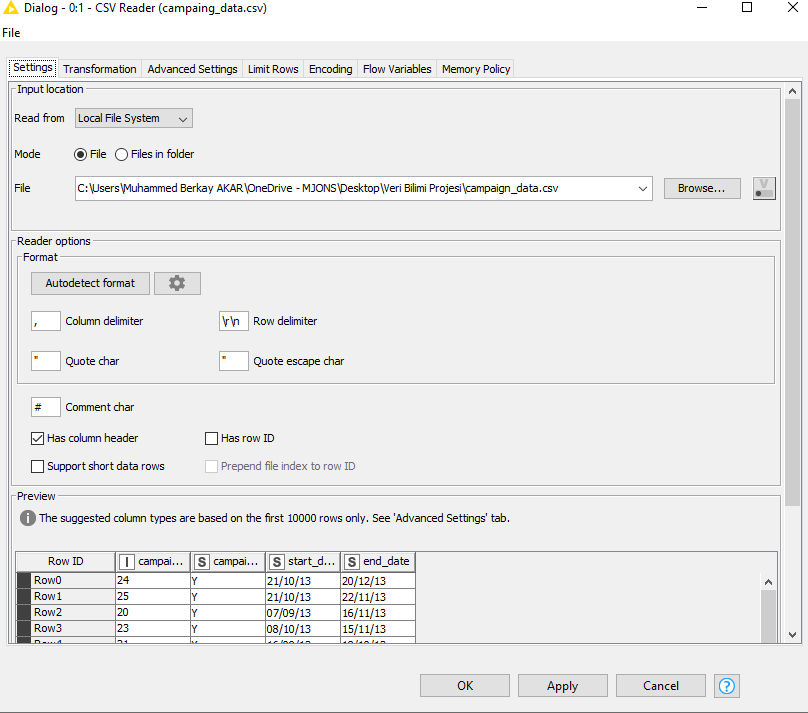
Çalışma boyunca aynı problem ve aynı veri seti kullanıldığı için bu bölümde ağırlıklı olarak makale çalışmasının Knime Analytics Platform’u üzerine uygulanması ağırlıklı olarak amaçlanmıştır.



Şekil 20 Çalışma şablonu (Workflow)

**9.1. Verilerin Okunması:**

Knime workflow olarak adlandırılan verilerin uçtan uca gerekli düğümler üzerinden işlemlere uğrayarak aktığı ve sürekli olarak bir akışın gerçekleştiği bir yazılımdır. Python üzerinde pandas kütüphanesi altından read\_csv() komutu kullanılarak yapılan işlemler bu aşamada herhangi bir kod yazımı gerektirmeden ilgili düğüm üzerindeki ayarların düzenlenmesi ile gerçekleştirilecektir. Okunacak dosya tipine göre gerekli düğümün seçilmesi gereklidir. Bu aşamada veri setindeki verilerin hepsi csv dosya tipine sahip olduğundan ötürü IO/Read/CSV Reader düğümünün kullanılması gerekmektedir. 5 farklı veri setinin olmasından ötürü 5 adet düğüm kullanılmıştır ve ilgili düğümlerin açıklamaları altlarında mevcuttur. Farklı veri tipindeki dosyalar için IO/Read yolunun altında ilgili dosya tipine göre okuyucular mevcuttur.



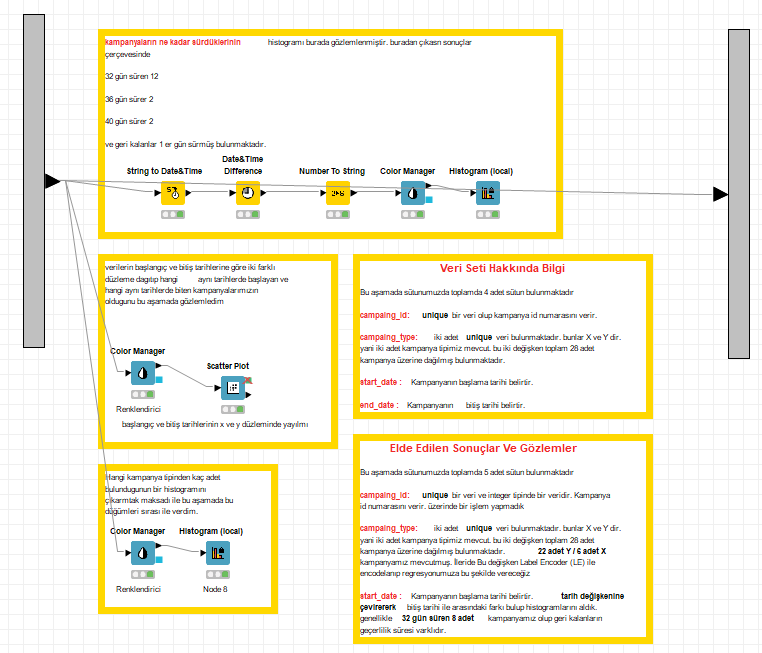
Şekil 20 CSV Reader düğümünün ayarları

File bölümünün yanındaki browse kısmı üzerinden okunacak veri setleri tek tek seçilebilmektedir. Beraberinde csv dosyası içerisindeki satır ve kolon ayıraçları format sekmesi altında gerekli şekillerde elle verilebildiği gibi AutoDetect format butonu ile düğümün otomatik olarak verileri ayraçlarının bulunması gerçekleştirilebilmektedir. Beraberinde her ilgili düğüm altındaki açıklamaya göre veri setlerinin tek tek okunması ile gerçekleştirilmiş bulunmaktadır.

**9.2. Veriyi Tanıma Metanodeları**

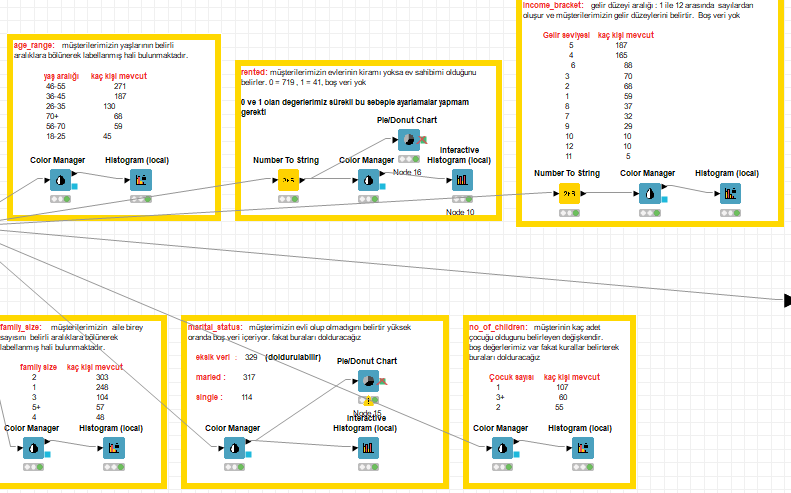
Veri biliminin en önemli aşamalarından birisi olan veriyi tanımak aşamasında yapılan keşifçi veri analizi yöntemlerinde içerisinde birçok grafikler bulundurması sebebi ile ana veri akışında kirlilik ve karmaşıklık yaratmaması ve anlaşılmasının kolaylaştırabilmek için Knime içerisinde Metanode olarak isimlendirilen kapsayıcı ve kendi içerisinde birden fazla düğümün çalışmasına izin veren yapıların daha küçük bir yapı içerisine işlemleri gerçekleştirerek akışta temiz ve anlaşılırlık sağlanmıştır. Her bir tablo kendisi ile ilgili bir metanode a bağlanarak işlemler gerçekleştirilmiştir.

1. **Campaing\_data Metanode’u:** bu metanode campaing data ya ait olan verilerin görselleştirme işlemlerinin belirtildiği metanode dur. İlgili metanode içerisinde kampanyaların kaç gün sürdüğünü belirten bir histogram grafiği, kampanyaların başlangıç tarihlerinin bitiş tarihlerine göre dağılımını belirten bir scatterplot grafiği, ve hangi kampanya tipinden ne kadar bulunduğunu işaret eden bir histogram grafiği mevcuttur. İlgili grafiklerin renklendirilmesi için Color Manager düğümü, scatter plot grafiği için views/Javascript/Scatter plot düğümü ve histogram grafikleri için /Views/local altındaki histogram grafikleri kullanılmıştır. Tarihlerin histogramı için öncelikle String to Date Time düğümü ile zaman dönüşümü gerçekleştirmesi için Date&Time Difference ile başlangıç ve bitiş tarihlerinin arası sayısallaştırılıp gün bazında değerleri alınmıştır.



Şekil 21 Campaing\_data Veriyi Tanıma Metanode’u

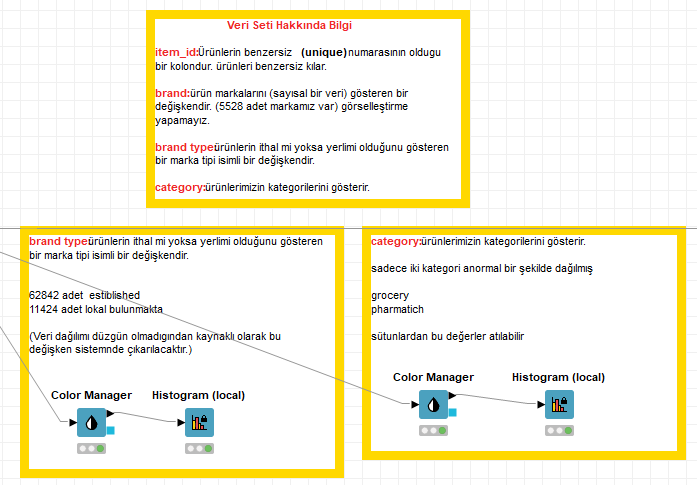
1. **Coupone\_İtem\_Mapping Metanode’u:** Metanode’un içerisinde 2 adet Groupby düğümü bulunmaktadır. Amaçları ürünler ile kuponların eşleşmesi içeren listeleri ve hangi kupona ne kadar itemin (item sayısı) eşleştiğinin tespitini yapmak içindir. Herhangi bir görselleştirme içermez.
2. **Costumer\_Demograpich Metanode’u:** Bu metanode müşteri demografik özelliklerini barındıran veri seti ile ilgili görselleştirme ve veri anlaşılması için gerekli grafikleri barındıran bir metanode’dur.

****

Şekil 22 Cosumer Demograpich Veriyi Tanıma Metanode’u

İçerisinde aile sayısının histogramı, evlilik durumunun renklendirilmiş bir biçimde pasta ve histogram grafikleri, çocuk sayısı histogramları, gelir seviyeleri ve bu gelir seviyesine ait ne kadar müşterinin olduğunu içeren histogram grafikleri, rented statüsü ve yaş aralıkları, bu yaş aralığına ne kadar kişinin dahil olduğununu belirten gistogram grafiğine yer verilmiştir.

1. **Costumer\_Transaction Metanode’u:** Bu metanode müşteri işlemlerinin görselleştirmesi ile ilgili bir metanode dur bu aşamada veri seti hakkında bilgilendirme yapılmış olup herhangi bir görselleştirme yapılmamıştır.
2. **İtem\_Data Metanode’u :**  Bu metanode ürünlerin verileri hakkında bilgilendirme yapan bir metanodedur. Gerekli bilgilendirme ve marka , kategori histogram grafikleri bu aşamada oluşturulmuştur.

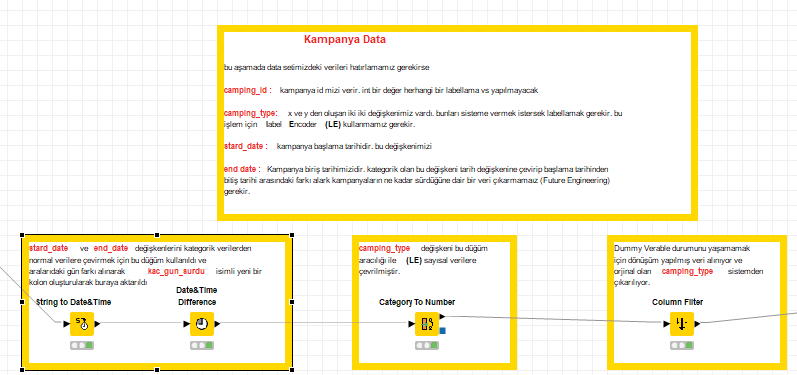
****

Şekil 23 İtem\_Data Metanode'u

**9.3 Ön işleme (Preprosessing)**

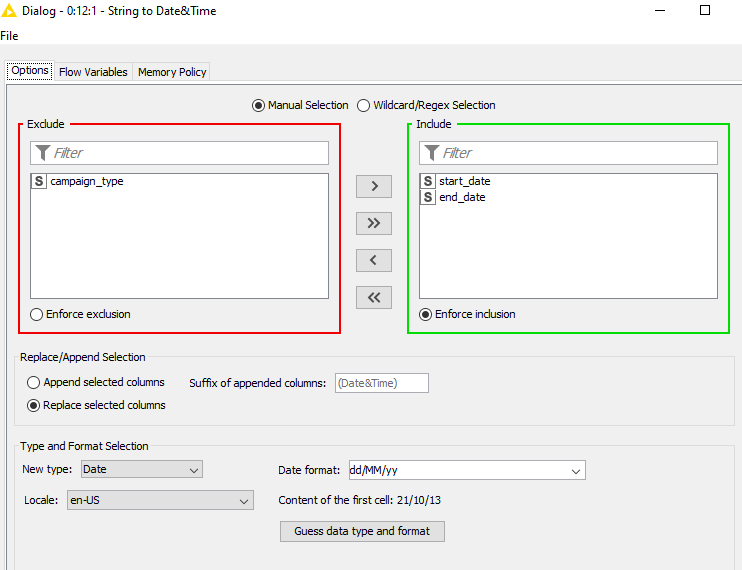
Python kısmında uygulamış olduğumuz ön işleme aşamaları burada da uygulanacaktır. Her bir veri setinin kendisine ait bir ön işlemesi olacağından ötürü veriyi tanıma metanodeların dan çıkan veriler kendisine ait olan ön işleme metanodelarına gelerek işlemlere uğrayacaktır.

1. **Kampanya Dataları Ön İşleme Metanode’u**



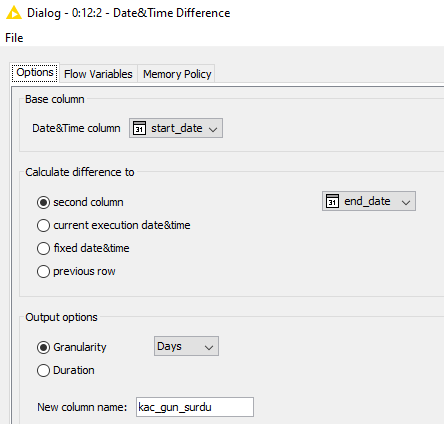
Şekil 24 Kampanya Data Ön İşleme Metanode'u Akış Şeması

Tarih değişkenleri sistemde kullanılabilmesi ve tarihler arasındaki günün hesaplanabilmesi için String To Date & Time Node’u kullanılmaktadır. Yapılan işlem zaman değişkeni stringten formatından tarih formatına bu şekilde çevrilebilmektedir. Çevrilecek olan değişkenlerin include içerisine alınması ve Date format kısmından gelen stringin hangi formatta olacağı belirlenmesi gerekmektedir.



Şekil 25 String To Date&Time ayarlaması

Beraberinde tarihler arasındaki farkı hesaplayabilmek için çıkan zaman formatındaki değişkenleri Date&Time Difference düğümüne aktarılmadı ve içerisinde seçim yapılmalıdır.



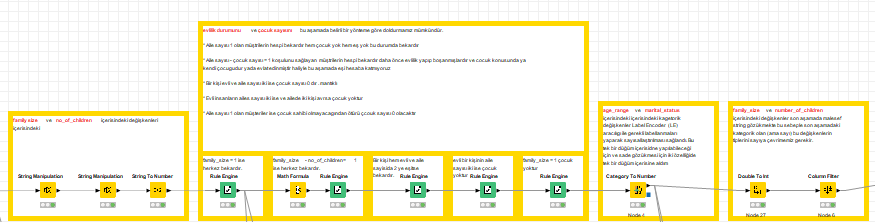
Şekil 26 Date&Time Difference Düğümü

Çıkan kampanya tipi değişkenlerinin ordinal bir biçimde sıralandırabilmemiz için LabelEncoding işlemi uygulamamız gerekmektedir. Category To Number kullanım düğümünün amacı budur. Bu sayede tahmin algoritmalarında veriler string formattan sayısallaştırmalar yapılarak kullanılabilmektedir. Bu dönüşüm sonrasında string formattaki kampanya tipi içeren değişkenin kullanılmaması gerektiğinden ötürü bir Columb filter düğümü vasıtası ile bu değişkeni burada filtrelenmektedir.

1. **Kupon İtem Eşleşmesi Ön İşleme Metanode’u:**

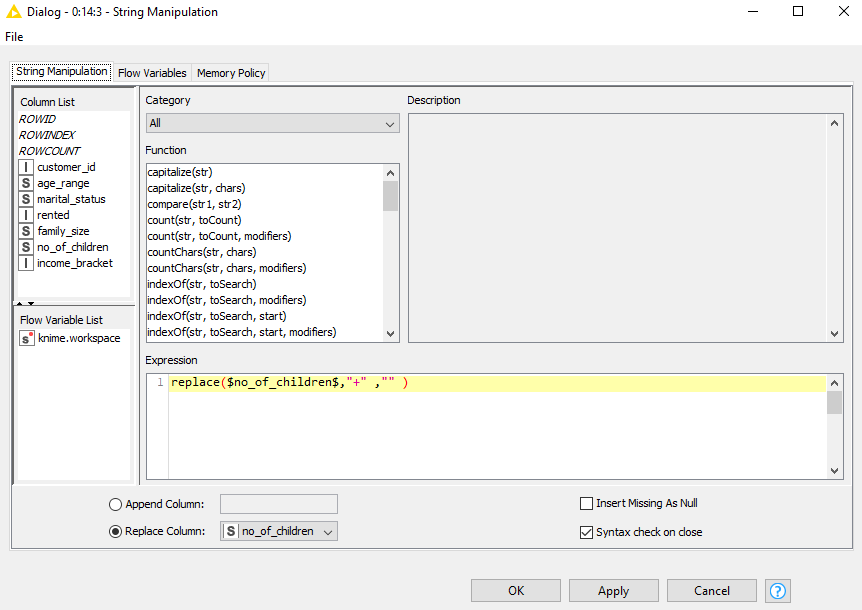
Veri setinde gözüktüğü üzere birden kuponun bir üründe ve bir kuponun birden fazla ürün üzerinde eşleşmesi sebebi ile bu aşamada herhangi bir ön işlemeye gerek yoktur. Beraberinde bu veri seti kullanılmayacaktır.

1. **Müşteri Demografik Özellikleri Ön İşleme Metanode’u:**

****

Şekil 27 Müşteri Demografik Özellikleri Ön İşleme Metanode'u

Family size değişkenindeki ve çocuk değişkenindeki + değişkenlerinin kaldırılması için String Manipulation düğümü kullanılmıştır. Bu düğüm veriler üzerinde birçok method yardımı ile istenilen değişikliklerin yapılmasını sağlayan bir düğümdür. Kullanım örneği aşağıda verilmiştir.

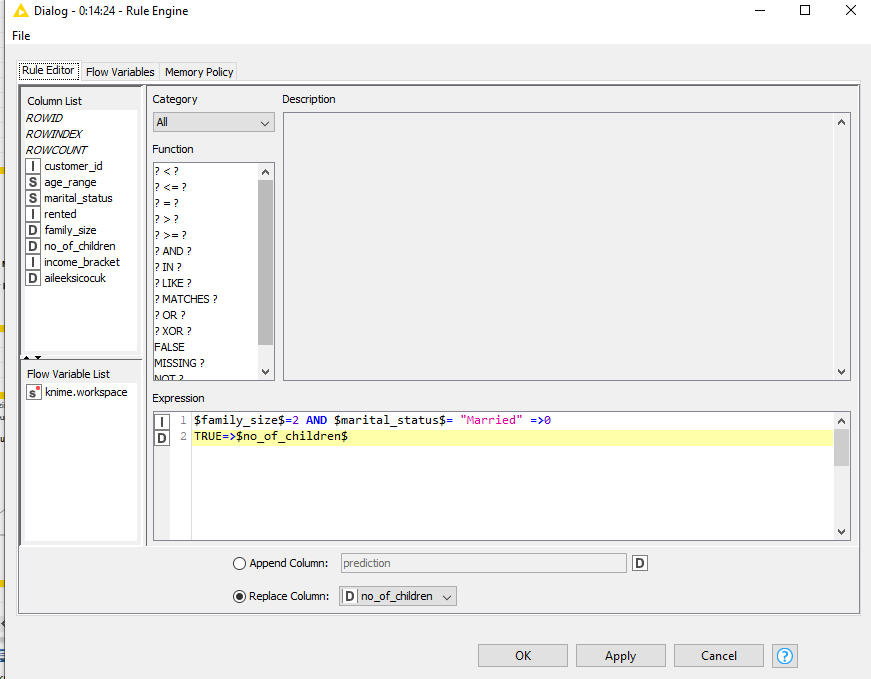


Şekil 28 String Manipluation düğümü ayarları

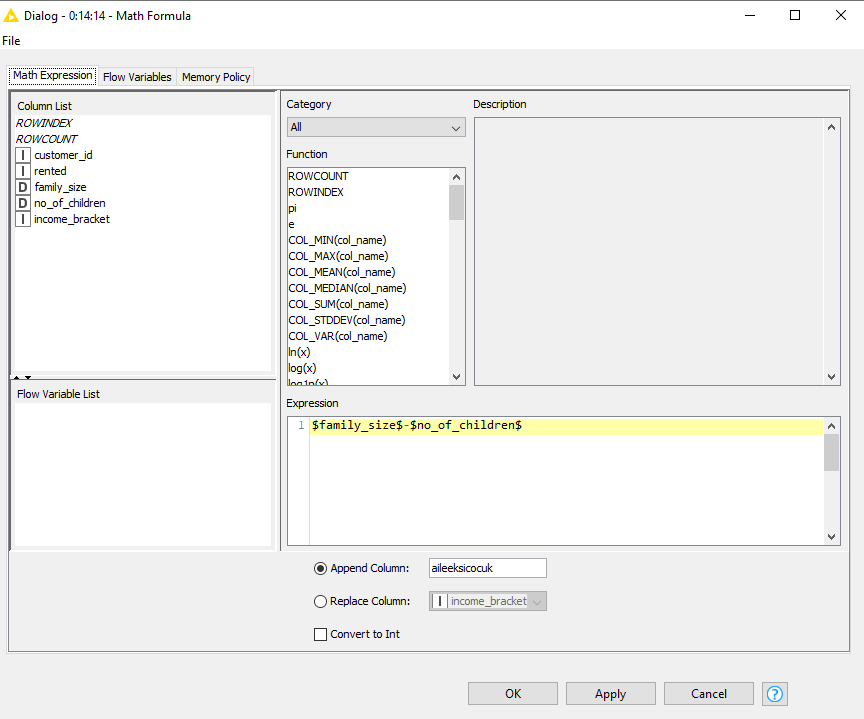
Üstteki verilen örnek sadece çocuk sayısı düğümü için gereçli iken bu aşamada aile boyutu içinde uygulanmıştır. Verileri sayısallaştırabilmek adına Category to Number değişkenide bu aşamada kullanılmıştır. Python kısmında yapıldığı gibi bu aşamada eksik değerleri değişkenin kullanılabilmesi için doldurulması gerekmektedir. Burada doldurma işlemlerini belirli bir kural tabanına göre gerçekleştirilecektir. Kurallar python ile geliştirme kısmında belirtildiği gibi doldurulmuştur.

* Aile sayısı 1 ise kişi bekardır.
* Aile sayısı – çocuk sayısı = 1 ise kişi bekardır.
* Kişi hem evli hemde aile sayısı 2 ise bekardır.
* Evli bir kişinin aile sayısı 2 ise çocuk yoktur.
* Family\_Size=1 ise çocuk yoktur.

İşlemler için sütunlar arasındaki matematiksel işlemler Math formula düğümü ile gerçekleştirilmiştir. Bu düğümden çıkan nicelikler bir sonraki kural ile doldurmada kullanılacak olan Rule Engine isimli düğüm ile gerçekleştirilmiştir. İşlemler sonucunda evlilik durumunu ve yaş aralığı değişkenlerini ordinal bir şekilde sıralı ve sayısal olarak kullanabilmemiz için bir LabelEncoding işlemi gerçekleştirmemiz gerekmektedir. Bu işlemin karşılığı Knime platformundaki karşılığı Category to Number düğümüdür. Dönüşümde sağlandıktan sonra modelleme aşamasına geçilmiştir.

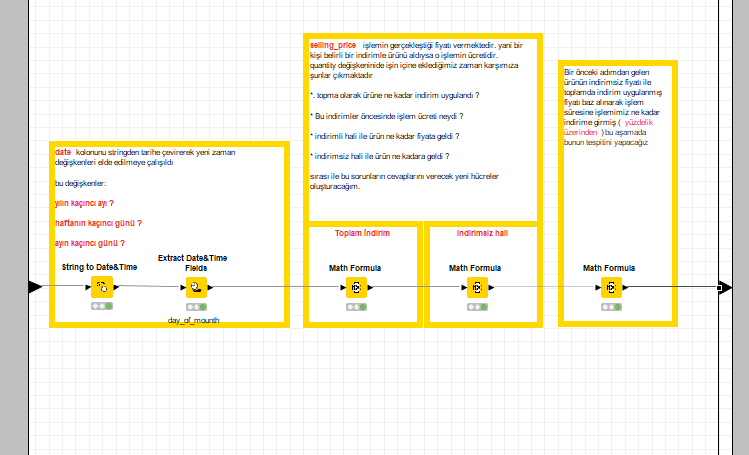


Şekil 29 Rule Engine düğümü kullanımına bir örnek



Şekil 30 Math Formula düğümü kullanımına bir örnek

1. **Müşteri İşlemleri Ön İşleme Metanode’u:**

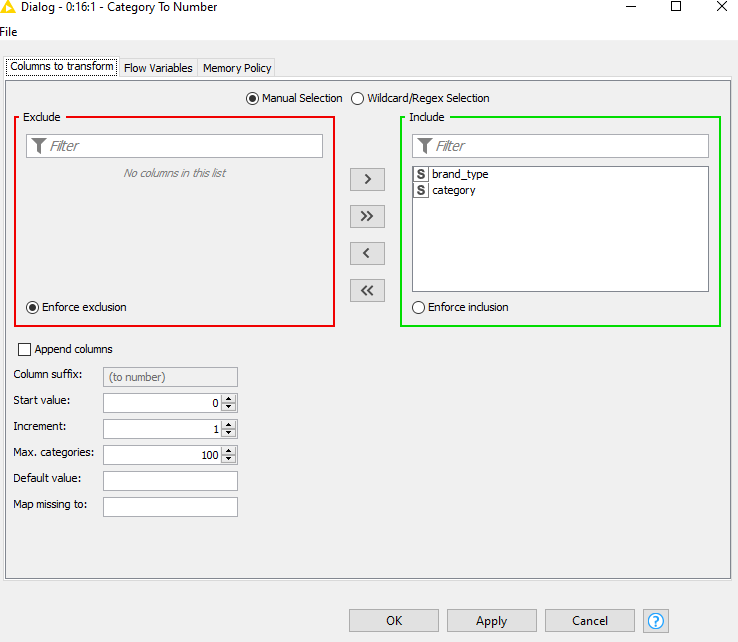
****

Şekil 31 Müşteri İşlemleri Ön İşleme Metanode'u

Müşteri işlemleri önişleme metanode’u içerisindeki gerekli ön işlemeler gerçekleştirilecektir. İlk olarak işlemler geçekleştirebilmek adına zamandan yeni özellikler (ayın günleri, yılın ayları ve haftanın günleri gibi) çıkarabilmemiz için string formatta gelen değişkenimizi tarih formatına String to Date&Time düğümü ile gerçekleştirmemiz gerekmektedir. Beraberinde ürünlerin indirimsiz fiyatlarını bulabilmemiz için indirim oranlarını toplayıp mutlak değer içerisine alarak ürünün indirimsiz fiyatlarını bulmamız gerekmektedir. Toplam indirim değişkenlerini bularak son aşamada yüzdelik olarak bir dönüşüm yapmamız gerekmektedir.

1. **Ürün Özellikleri Ön İşleme Metanode’u:**

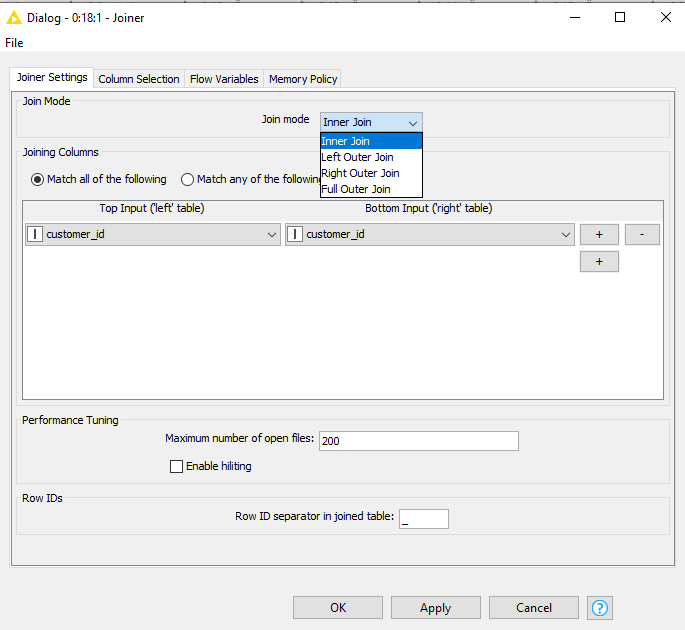
Ürün özelliklerini tablosunun ön işleme süresince sadece kategorik olan değişkenlerimizi nümerik olarak çevrimini yaparak LabelEncoding yöntemi kullanılmıştır. Bu dönüşüm için Category To Number düğümü kullanılması gerekmektedir.



Şekil 32 Category To Number düğüm ayarlaması

**9.3 Modelleme**

Modelleme aşamasına geçmeden önce yapmamız gereken bazı işlemler bulunmaktadır. Veri setlerinini gerekli anahtar değerlerine göre birleştirmeler kullanarak birleştirmeler yapmamız gerekmektedir. Süreç içerisinde birleştirmemiz gereken 3 adet tablomuz mevcuttur. Customer\_transaction, customer\_demograpich ve item\_data isimli veri setlerinin anahtarlar vasıtası ile birleştirmeleri gerekmektedir. İlişkili tabloların birleştirilmesi metanode içerisinde bu işlemler gerçekleştirilmiş bulunmaktadır. Beraberinde birleştirilme aşamalarından bahsetmek gerekirse ilk aşamada Joiner düğümü kullanılarak customer\_demograpich ve customer\_transaction tabloları customer\_id isimli değişken vasıtası ile inner join metodu kullanılarak birleştirme yapılmıştır. Yapılan birleştirmede 882 adet ortak olmayan müşterilerimiz silinmiş ve işlemler tablosundaki yaklaşık olarak 1.2 m kayıt 761 bine kadar düşmüştür. İkinci aşamada ise buradan gelen verilerin item\_id değişkeni aracılığı ile inner join kullanılarak bir birleştirilmesinin yapılması ve ardından çıkan değerlerin metanode çıkışına verilmesi gerekmektedir. inner join metodunu kullanabilmemiz için gerekli ayarlar aşağıda verilmiştir.

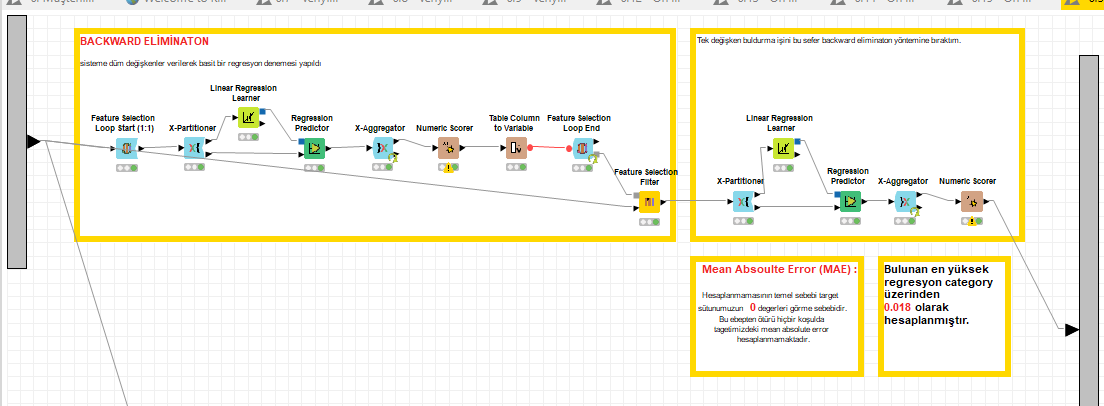


Şekil 33 Joiner düğüm ayarları

Gerekli ayarlamaların beraberinde algoritmaya verilecek olan veri setimiz 5 farklı algoritma kullanılarak değerlendirilmiştir. Simple Linear Regression, Multiple Linear Regression, Simple Regression Tree, XGBoost Regression Gradient Boosting Regression algoritmaları kullanılarak yapılan modellemerin detayları aşağıdaki başlıklarda belirtilmiştir.

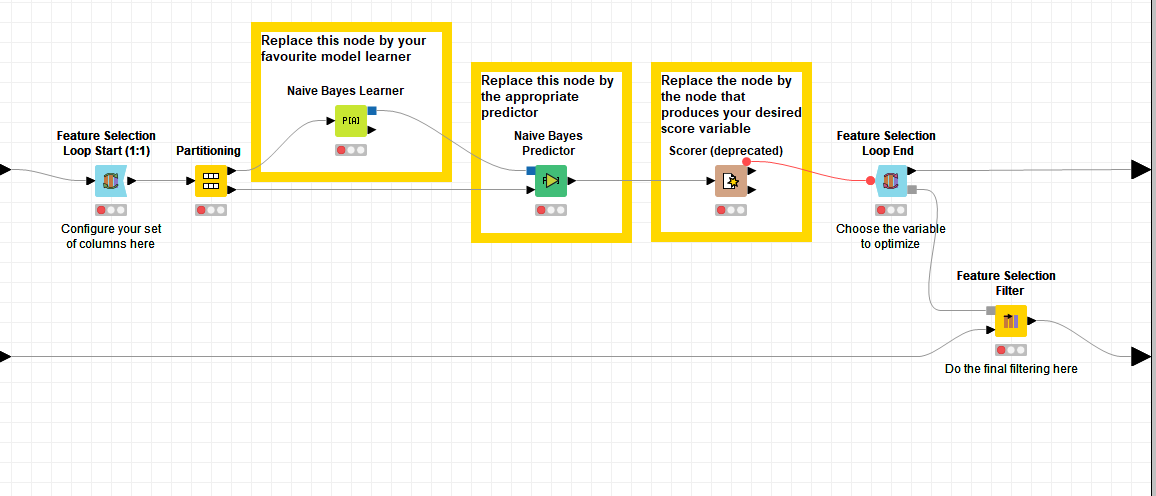
**9.3.1. Simple Linear Regression (SLR):**

Basit doğrusal regresyon olarak Türkçeye çevirebileceğimiz algoritmanın çalışma yapısından bahsetmemiz gerekirse bir bağımsız değişken üzerinden 1 bağımlı değişken tahmininde kullanılmaktadır. Kullanım süreci içerisinde yapılan analizler neticesinde değerler python kullanılarak yapılan değerlere benzerlik göstermektedir.



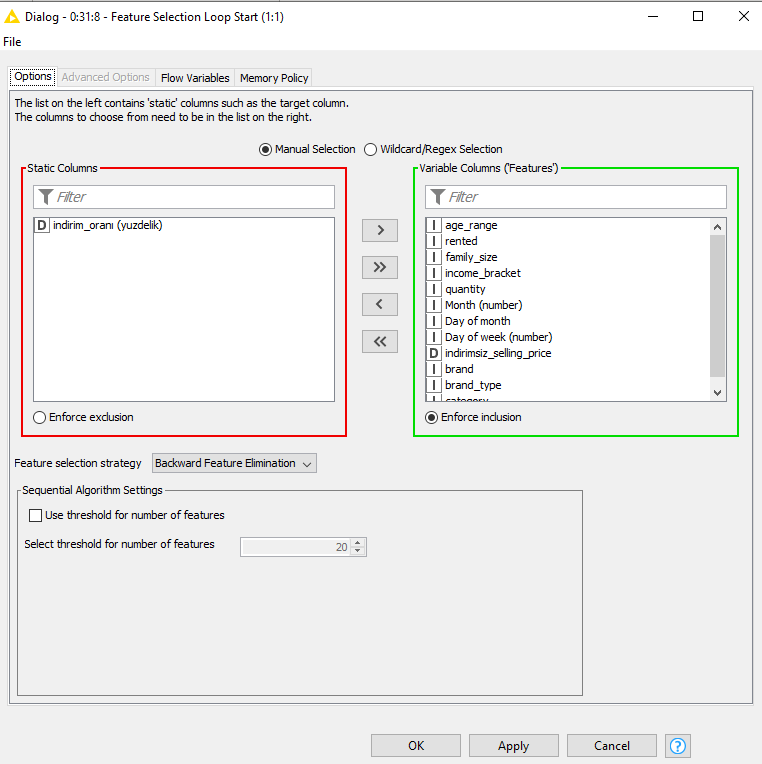
Şekil 34 Basit Doğrusal Regresyon Workflow şeması

Knime için değişken eleme yöntemleri arasında en çok kullanılan sarmal yöntemlerden birisi olan backward elimination yöntemi kullanılmıştır. Tarafımca tek tek elle edilen değişkenin doğruluğunu kanıtlayabilmem için bu yöntem ile en yüksek tek değişkenden elde edilen tahmin bulunması hedeflenmiştir. Knime platformu içerisinde Forward Feature Selection isimli bulunan bir metanode içeriği kullanılarak ve belirli değişiklikler yapılarak gerçekleştirilmiştir.

****

Şekil 35 Varsayılan olarak gelen forward feature selection metanodeu

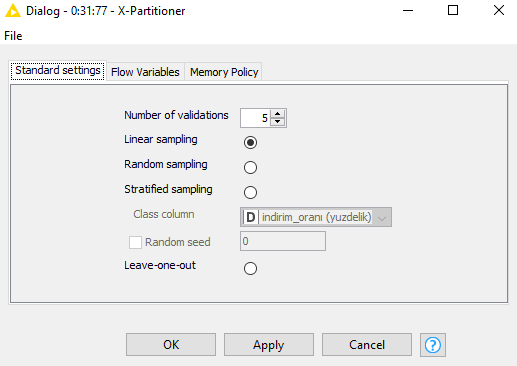
Şekil 35 te gösterildiği üzere varsayılan olarak gelen Forward Feature Selection metanode’u üzerinden açıklamalar gerçekleştirilmesi gerekirse döngü işleyişi için döngü başlatıcı düğümü olarak Feature Selection Loop Start kullanılması gerekmektedir. bu düğüm üzerinden tahmin edilmesi planlanan hedef değişkeni ana veriler üzerinden ayrıştırılmalı ve seçim yöntemimizi belirlemeliyiz. Ceri seti istenilen oranlarda eğitim ve test kümesi olarak bölünmesinin ayarlanması ile beraberinde knime döngü için istediğimiz algoritmayı seçmemizde esneklik sağlamaktadır. Yapılacak işe göre gerekli öğreniciler ve eğiticilerin ayarlanmasının ardından scorer üzerinden gerekli olan ölçüm metriği belirlenmesi ile birlikte döngü başlamaktadır. Feature Selection Loop end kısmı üzerinden döngüyü başlatmalı ve Feature selection Filter düğümü ile hesaplanan değişkenlerin oluşturduğu R2 değerine göre istediğimiz kombinasyonu seçerek tekrardan bir eğtim ve test sürecinde bulunmalı ve hesaplamalar yapmalıyız.



Şekil 36 Feature selection loop start düğüm ayarları

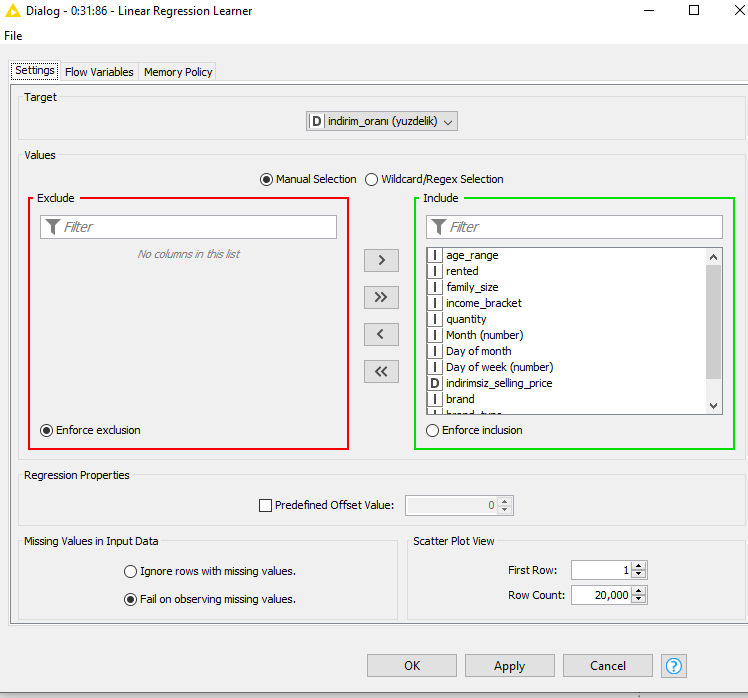
Yapının çalışmalarımıza uygulanması aşamasına gelindiğinde Feature Selection Loop Start ile gerekli düzenlemeler şekil 36 da gözüktüğü gibi gerçekleştirilmiştir. Hedef değişkenimiz indirim oranı olması sebebi ile static columns altına taşınarak stratejiyi bacward feature eliminaton olarak belirtilmiş bulunmaktadır.

Bir sonraki aşamada veri setimizin eğitim ve test aşamaları gerçekleşecektir. Bu gerçekleşme işi için tüm değişkenlerin sırası ile sisteme verilerek eğitilmeleri ve çıkan değerler çerçevesinde test edilmeleri gerekmektedir. bu şekilde tüm veri seti hem eğitilip hem test edildiği için doğrulama gerçekleştirilmiş olacaktır.



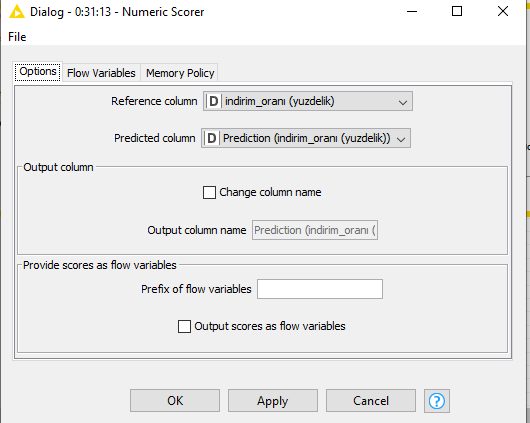
Şekil 37 X-Partitioner ayarları

Backward-eliminaton yöntemi boyunca Linear Regression algoritması kullanılmakta ve bu algortitma üzerinden tahminler gerçekleştirilmektedir.



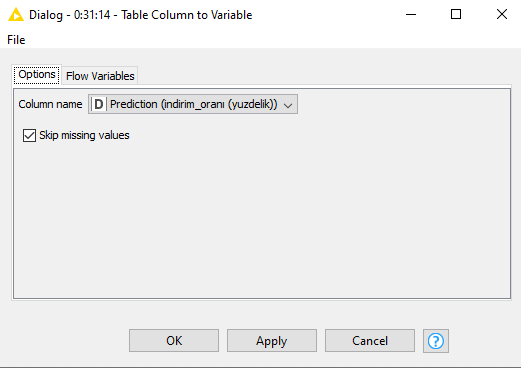
Şekil 38 Linear Regression Learner üzerinden yapılan ayarlar

Çıkan değerlerin bir skorlama düğümü vasıtası ile skorlanması gerekmektedir. Bu iş için birçok skorer düğümü mevcuttur. Tahmin algoritmalarında sıklıkla kullanılacak olan scorer düğümü ise Numeric Scorer isimli düğümdür. Skorlamak için hedef düğümümüz ve tahmin edilen değişken düğümlerini bu aşamada vermemiz gerekmektedir.



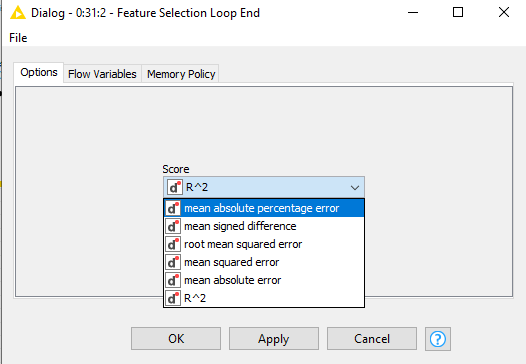
Şekil 39 Numeric Scorer ayarları

Değişkenlerimizi seçim için bir loop end düğümüne aktarmamız gerekmektedir. Knime üzerinde herhangi bir döngü içeren işlem yaparkan bu düğümün bir başlangıç ve bir bitiş düğümü kullanılması gereklidir. Başlangıç ve bitiş düğümleri yapılacak işlem döngüsüne göre özeldir. Bu aşamada değişken seçimi yapılacağından ötürü Feature Selection Loop End düğümü kullanılacaktır. Numeric Scorer dan gelen değerlerin tablo değeri olduğundan ve bu değerlerin bir döngüye aktarılması gerektiğinde Table Columns to Verable değişkeni ile bu dönüşümü gerçekleştirip döngü bitiş değişkenine aktarmamız gerekmektedir.

****

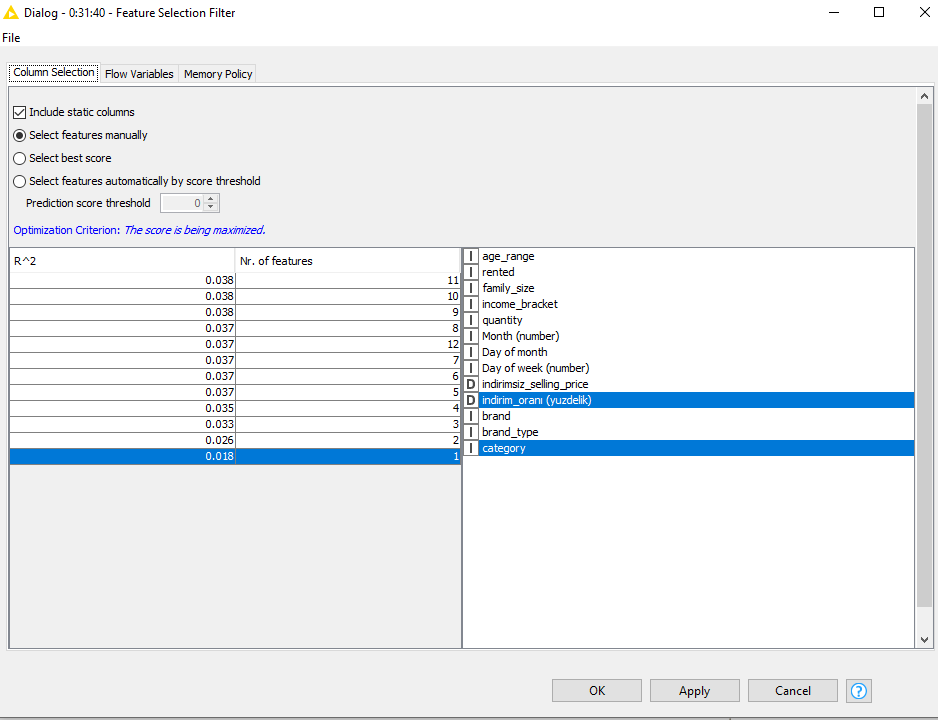
Şekil 40 Table Column To Verable değişkeni ayarlamaları

Şekil 41 de döngü bitişi ayarları görülmektedir. Seçilecek hesaplama metriğine göre seçimler yapılması ve bu düğümün başlaması ile döngü başlayacaktır. İşleyiş boyunca R2 metriği hesaplama metriği olarak kullanılmıştır.



Şekil 41 Feature Selection Loop End ayarları

Yapılan elemeler sonucunda Feature Selection Filter vasıtası ile özellik seçiminden gelen model giriş modeli olarak aktarılır ve bir üst aşamada değişkenlerin hepsini alabilmesi için giriş değişkenleri üçgen sembolü içeren değişken içerisine aktarılmalıdır.

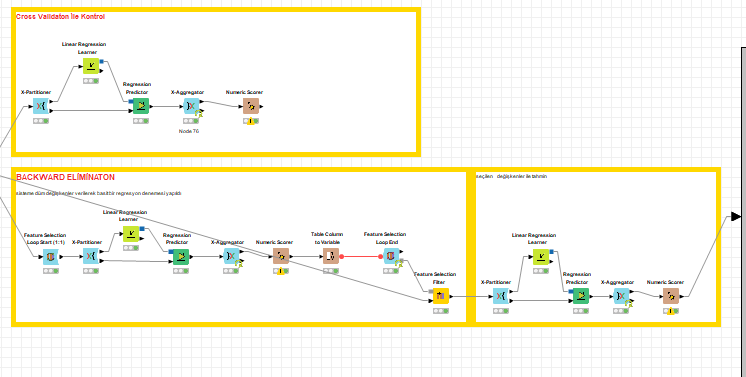


Şekil 42 Feature Seleciton Filter ayarları

Tek bir değişken üzerinden tahmin yapacağımız işin değişken sayımızı 1 olarak belirttik. İnclude static columns seçeneğini seçmemizin sebebi değişken filtreleme yaparken referans alınan kolonun sisteme dahil edilmesi gerekmektedir. Değişken seçimi yapıldığına göre son aşamada bu değişken üzerinden tahminlerimizin gerçekleştirilmesi gerekmektedir. Tahmin aşamasında şekil 34’te görüldüğü gibi verilerin train ve test olarak bölümlendirilmesi gerekmektedir. yapılan bölümlendirme için 5 ve linear sampling seçilerek tahmine başlanmıştır. Yapılan regresyon sonucu R2 değeri 0.018 olarak hesaplanmıştır.

**9.3.1. Multiple Linear Regression (MLR):**

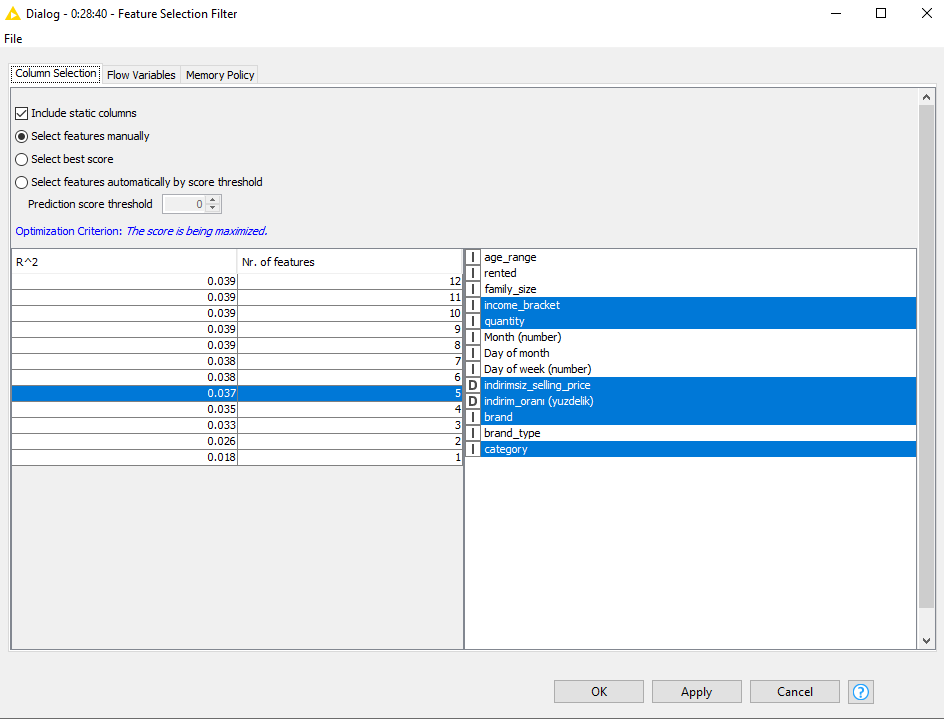
Çoklu değişken regresyonu olarak bilinen bu regresyon tipi basit doğrusal regresyondan farklı olarak bir bağımsız değişkenden üzerinden 1 bağımlı değişken tahmini olarak değil birden fazla bağımsız üzerinden bir bağımlı değişken tahmini yapmak üzerine kurulu olan regresyon tipidir. SLR ile MLR için Linear Regression Learner ve Linear Regression Predictor düğümleri kullanılması gerekmektedir.

****

Şekil 43 MLR için iş akışı örneği

İlk olarak tüm değişkenleri sisteme dahil ederek bir model denemesi yapıldı. Tüm değişkenlerin model kurma için etkilerini bu şekilde gözlenmek istenmiş olup bir denemedir. Düğüm akışları ve ayarları aynı özellikleri göstermesinden ötürü bu aşamada ve daha sonraki bölümlerde farklı olan işlemlerde görselleştirmeler yapılacaktır. Görselleştirme olmayan yerlerde SLR için yapılan ayarlar ile aynıdır. Çalışma uygulanmak istendiği taktirde gerekli ayarlamalar için SLR bölümü okuyucu için bir kılavuz oluşturacaktır.

Tüm değişkenlerin dahil edilmesi halinde elde edilen R2 değeri 0.037 olarak hesaplanmıştır. Değişken seçimi döngüsünün başlaması sonrasında olabilecek en az değişken sayısından kabul edilebilir değer elde edilmek amaçlanmıştır. Regresyon değerine fazla bir etkisi olmayan değişkenin sistemden çıkartılması modelleme süresince bizlere zaman tasarrufu sağlayacaktır.

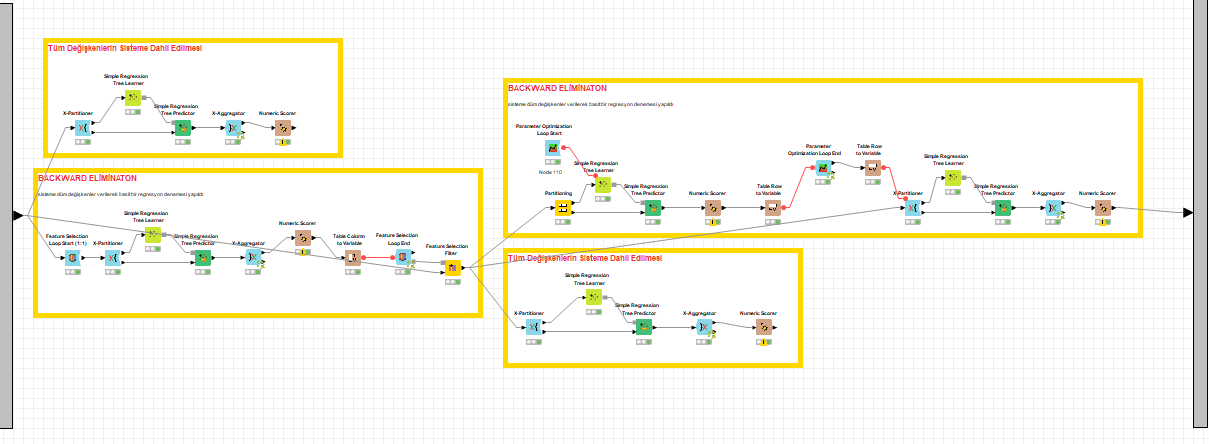


Şekil 44 MLR için seçilen değişkenler

Yapılan seçimler sonucunda cross validation sonucunda elde edilen son aşamadaki en optimize model üzerinden hesaplanan R2 değerimiz 0.038 olarak hesaplanmıştır.

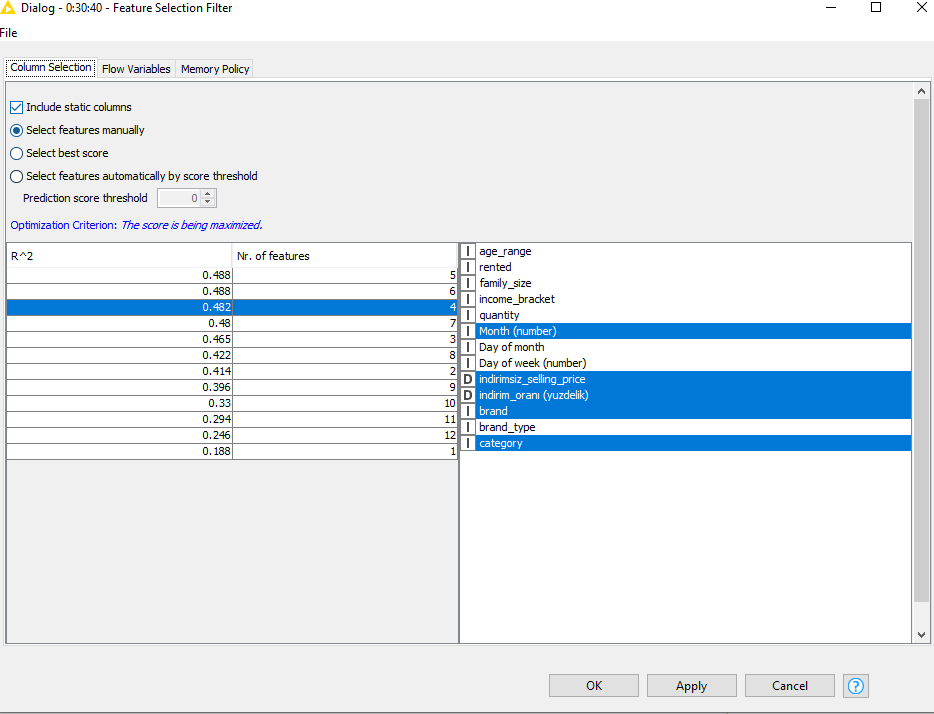
**Simple Regression Tree Regression (STR):**

Bu algoritma ağaç tabanlı bir regresyon algoritması olan bir regresyon tipidir. İşleyiş için workflow şekil 45 üzerinde mevcuttur.



Şekil 45 Simple Regression Tree algoritması için bir iş akışı

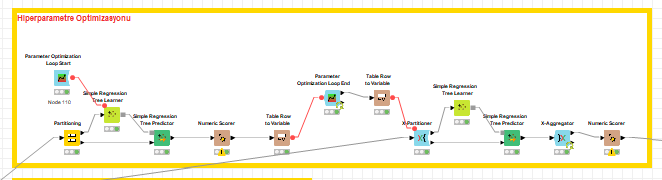
Tüm değişkenlerin sisteme eklenmesi ile elde edilen R2 değerimiz 0.073 olarak hesaplanmıştır. Değişken modellemesinde hem doğruluk değerinin artması hem de modelleme hızından ve performansından kazanç sağlayabilmek için değişken elemeye gidilecektir. Değişken eleme yöntemi Backward Eliminaton olup işlem için öğrenici ve tahminleyici düğümlerimiz Simple Regression Tree algoritmasına uygun olan öğretici ve tahmin edici ile değiştirilmiştir. Cross validaton değerimizin 5 seçilmesiyle birlikte elde elde edilen değişkenler şekil 46 da belirtilmiştir.



Şekil 46 Simple Regression Tree için değişken seçimi

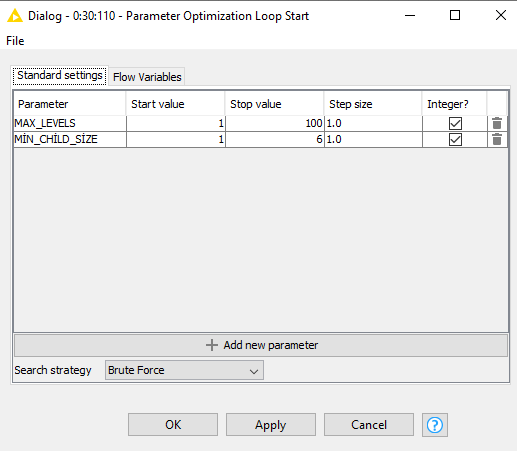
Elde edilen değerleriyle birlikte seçilen yeni değerler üzerinden cross validaton ile yapılan testler sonucu R2 değerimiz 0.442 olarak hesaplanmıştır. Modelin iyileştirilebilmesi için bu aşamadan sonra hiperparametre optimizasyonu yapılması gerekmektedir.

Knime platformunda hiperparametre optimizasyonu için 2 adet düğüm mevcuttur.Parameter Optimizaton Loop Start ve Parameter Optimization Loop End düğümleri hiperparametre optimizasyonu yapmamızı sağlayan iki adet düğümdür. Düğümler Knime içerisinde Knime Optimizaton isimli bir eklenti paketinin içerisinde mevcuttur. Platformun ilk indirilmesi ve yüklenmesinde bu eklenti paketi yüklü bulunmamaktadır. Gerekli eklenti yüklenmesi taktirinde bu eklenti kullanılabilir olmaktadır.



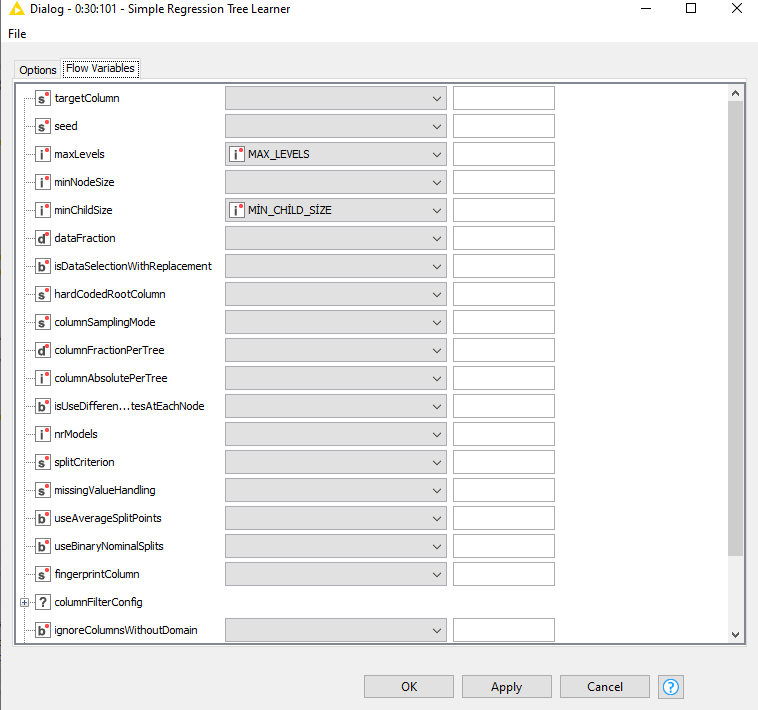
Şekil 47 Hiperparametre optimizasyonu aşaması için bir iş akışı örneği

İşlemin gerçekleştirilmesinde ilk aşama hiperparametre optimizasyonu için model parametrelerinin belirlenmesi ve aralıkların tanımlanmasıdır. Yapılan tanımlamalar neticesinde bir ağaç tipi olan bu regresyon için 2 adet parametre belirlenmiştir.



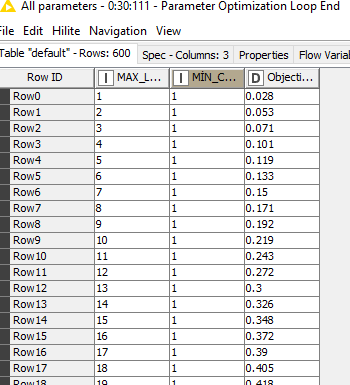
Şekil 48 Hiperparametrelerin tanımlanması

Tanımlama gerçekleşmesi ile birlikte arama stratejisini belirleyebiliriz. Tüm değişkenlerin denenmesi için Brute Force methodu kullanılmıştır. Başlangıç, bitiş ve arttırma değerleriyle birlikte değişken tipide belirtildikten sonra bu değişkenlerin modellerde kullanılması için ilgili öğrenme ve test etme algoritması düğümleri üzerinde gerekli ayarlamaları yapılması gerekmektedir. Yapılan ayarlamalar şekil 49’da görüldüğü gibidir.

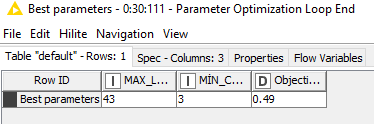


Şekil 50 Algoritmanın işleyişine devam edebilmesi için flow verable sekmesi altından gerekli düzenlemelerin yapılması

Yapılan ayarlamalar sonucunda Numeric Scorer’dan çıkan değerleri Table row to verable değişkeni ile birlikte Parameter Optimizaton Loop End kısmına aktarılması gerekmektedir. Parameter Optimizaton Loop End düğümü giriş değişken olarak sadece değişken kabul etmektedir. Tablo olan değişkenleri akış değişkeni olarak yapılandırmamız gerekmektedir. Parameter Optimizaton Loop End düğümünün iki adet çıktısı mevcuttur. Birinci çıktı best parameter olarak adlandırılan ve araştırma sonucu en iyi parametrelerin hangilerinin olduğunun tespitinin yapıldığı, ikinci çıktı ise tüm denemelerin ve sonuçlarının barındırıldığı bir yapıdır.



Şekil 50 tüm parametreler ve R2 değerleri

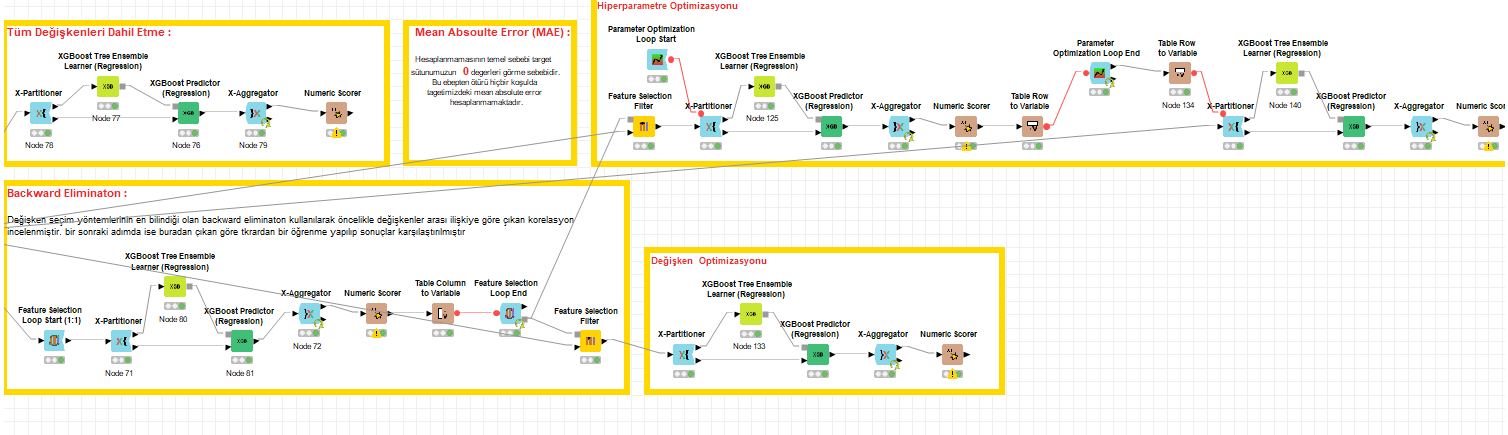


Şekil 51 best parameters altındaki değişkenler ve değerleri

Yapılan incelemeler neticesinde maksimum katman 43, minimum çocuk yaprak sayısı 3 bulunarak yapılan değerlendirmelerde R2 değeri 0.49 olarak hesaplanmıştır. Uygunluğun test edilebilmesi için çıkan değerlerin cross validaton kullanılarak tekrar değerlendirilmesi gerekmektedir. Yapılan tetkikler sonucunda R2değerimiz 0.46 olarak hesaplanmıştır.

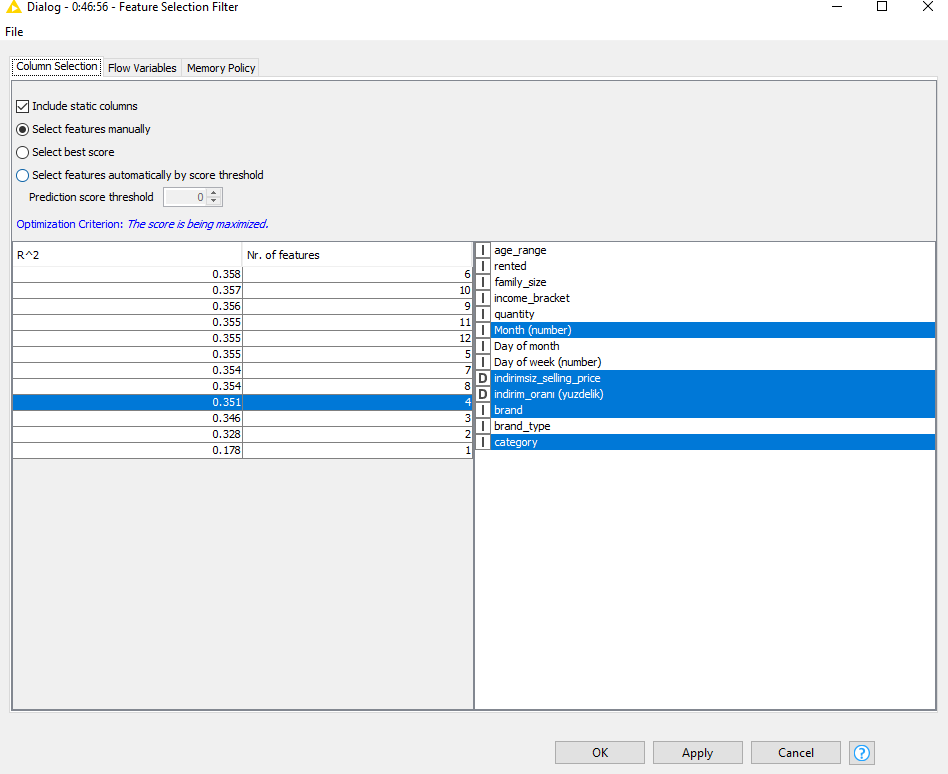
**Extreme Gradient Boosting Regression (XGBoost):**

Daha öncesinde detayları ile python kısmında anlatılan algoritmanın Knime kısmındaki uygulaması bu bölümde gerçekleştirilecektir. XGBoost algoritması kurulum beraberinde gelen bir algoritma olmayıp Knime Xgboost İntegration eklentisi ile kurulumunun gerçekleştirilmesi sonucu erişebileceğimiz bir ön koşul içerir. Gerekli eklentinin indirilmesi ardından tüm iş akışı şekil 52 de belirtilmiştir. İş akışı boyunca XGBoost Tree Regression algotirması kullanılmıştır.



Şekil 52 XGBoost iş akışı

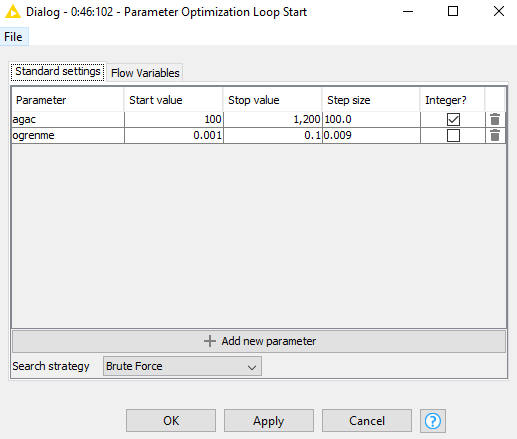
Tüm değişkenlerin sistem içerisinde dahil edilmesi ile ortaya çıkan R2 değerimiz 0.355 olarak hesaplanmıştır. Beraberinde modelin daha kararlı ve performanslı çalışabilmesi adına değişken seçimi gerçekleştirilmiştir. Diğer regresyon tiplerinde görüldüğü gibi değişken seçimi aşamasında regresyon algoritmaları XGBoost Tree Learner ve XGBoost Tree Predictor olarak değiştirilmiştir.



Şekil 53 XGBoost Tree Regression için yapılan değişken seçimi

Bu aşamada en az değişken ile hem performans hemde tahmin değerleri açısından şekil 53 te gözüken değişkenler tercih edilmiştir. Elde edilen değişkenler ile cross validaton doğrulaması kullanılarak yapılan hesaplar neticesinde R2 değerimiz 0.345 olarak hesaplanmış bulunmaktadır. Modelin daha kararlı çalışabilmesi adına bir sonraki aşamada hiperparametre optimizasyonuna girmesi gerekmektedir.

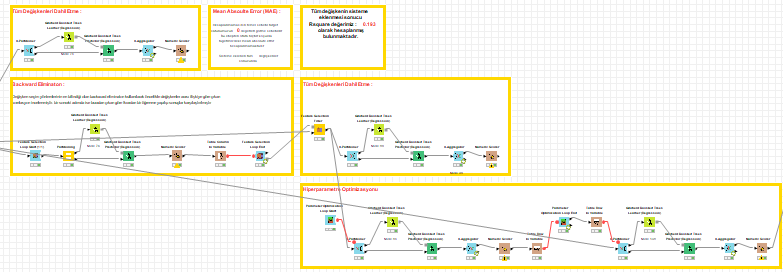
XGBoost Tree algoritması bir boosting algoritması olmasıyla birlikte tıpkı random forest algoritmas içerisinde olduğu gibi alt ağaçlar içermektedir. Beraberinde regresyon sonuçlarını en etkileyecek model parametreleri olarak n\_estimarots ve learning rate seçilmiştir.



Şekil 54XGBoost algoritması için seçilen hiperparametre ve arama startejileri

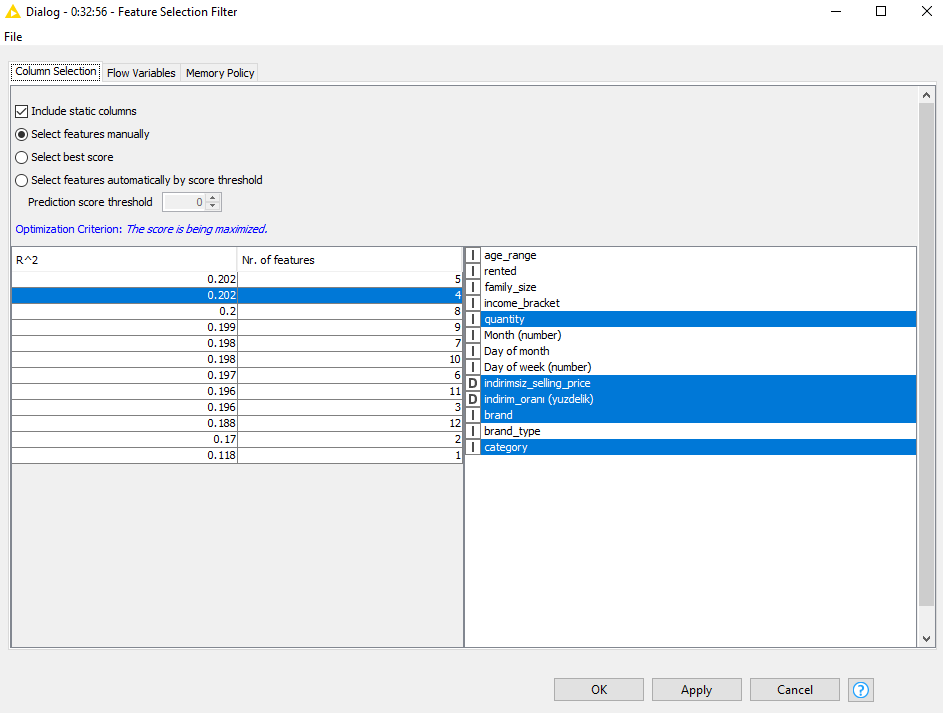
Değişkenlerin öğrenme düğümü içerisinde boostingRounds ve eda ya karşılık olarak belirlenmeleri ve akış değişkenlerinin kontrolü bu şekilde gerçekleştirilmelidir. Yapılan hiperparametre optimizasyonu sonucu en iyi öğrenme oranının 0.1 ve ağaç sayısının 1200 olarak belirlenmesiyle birlikte bu değerleri alarak cross validaton yöntemiyle beraber tahminleme de bulunulmuştur. Çıkan R2 değerimiz 0.46 dır.

**Gradient Boosting Regression (GBR)**



Şekil 55 Gradient Boosting Regression iş akışı

Python kısmında detayları ile anlatılan Gradient Boosting Tree algoritmasının iş akışı gözüktüğü üzere şekil 55 te verilmiştir. İşleme ilk olarak tüm veri setini vererek başlanılmıştır. Tüm veri setini vererek elde edilen R2 değerimiz 0.191 olarak hesaplanmıştır. Beraberinde diğer algoritmalarda da yapıldığı gibi backward eliminatoın yöntemi için gerekli olan tahmin ve öğrenme algoritmalarının değişikliğinin ardından yapılan testler sonucunda elde edilen değişkenler şekil 56 da gözüktüğü gibidir.



Şekil 56 Gradient Boosting Tree değişken seçimi

Değişken seçiminin ardından yapılan model doğrulamasında 0.202 olarak hesaplanmıştır. Modelleri daha stabil hale getirilebilmesi açısından hiperparametre optimizasyonu yapılması gerekmektedir. Gradient Boosting Regression algoritması Knime platformu için n\_estimators parametresine karşılık gelecek olan BoostingRonuds değişkeni bulunmamaktadır. Hiperparametre optimizasyonunda eat olarak adlandırılan öğrenim oranı parametresi değiştirilerek son aşamada gelinen değer 0.192 olarak hesaplanmıştır.

**10.) Özet**

Yapılan çalışmalar neticesinde bulunan çıkarımlar bu alanda özetlenerek tekrarı gerçekleştirilecektir. Müşteri davranışları altında müşterileri ve ürünleri üzerindeki gerçekleştirdikleri davranışlar analiz edilerek gerekli istatiksel modellemeler vasıtası ile müşteriye özgü indirim tavsiyesi uçtan uca veri bilimi süresince açık kaynak kodlu olarak Knime platformu ve python dilinin sklearn kütüphaneleri kullanılarak geliştirmeler tamamlanmış ve tüm veri setine uydurulabilecek şekilde gerekli yapılandırmalar yapılmıştır. Beraberinde alışagelmişin dışında her ne kadar regresyon değerleri düşük gelse dâhi İstatiksel açıdan gerekli çalışmaların izlenmesi, süreçlere uyum ve iş işleyişi denenen yöntemlerden yapılması gereken birçok çıkarım vardır. Elimizdeki veri setinin büyüklüğü ve istatiksel açıdan yeterli elverişliliğe sahip olmaması sebebi ile regresyon değerleri her ne kadar yüksek çıkmasa da iş işleyişi boyunca sürece sadık kalınmış ve küçük veri setleri üzerinde ezberlemeler yapılması yerine gerçek hayat senaryosu olarak adlandırılabilecek komplekslikte bir veri seti modellemesi tercih edilmiştir. Problem tipi her ne kadar regresyon problemi dahi olsa verilerin dengesiz dağılımlarından kaynaklı olarak lineer bir modelleme gerçekleştirilememiştir. Beraberinde iş için en iyi regresyon algoritmalarımız olarak ensemble learning altındaki algoritmalarımızla gerekli ve tatmin edici değerler yakalanmıştır. Sektörel manada düşünüldüğünde ise doğru tahmin değerleri kadar yapılan işleyişin hızı da önemli olup iki faktörü sağlayan algoritmalar seçilmelidir. Uçtan uca veri bilimi süresince geliştirilmiş olan tüm algoritmalar ve regresyon değerleri aşağıdaki tabloda belirtilmiştir.

|  |  |
| --- | --- |
| **Modeller** | R2 |
| Simple Linear Regression (Python) | - 0.015 |
| Simple Linear Regression (Knime) | 0.018 |
| Multiple Linear Regression (Python) | 0.027 |
| Multiple Linear Regression (Knime) | 0.038 |
| Desicion Tree Regression (Python) | 0.280 |
| Simple Regression Tree (Knime) | 0.469 |
| Random Forest Regression (Python) | 0.414 |
| XGBoost (Python) | 0.318 |
| XGBoost (Knime) | 0.460 |
| Gradient Boosting Regerssion (Python) | 0.232 |
| Gradient Boosting Regerssion (Knime) | 0.192 |
| LightGBM (Python) | 0.300 |

**11.) Sektörel Olarak Bilinen Bir Araç Üzerinden Skorlamalar**

****

AutoML yaklaşımı veri bilimi faaliyetlerini veri bilimcisine ihtiyaç duymadan veya daha az ihtiyaç duyarak yapabilmeyi amaçlayan verileri belirli yazılımlar vasıtası ile otomatik olarak analiz edebilen ve analizler beraberinde değerler üretebilen bir yöntemdir. Gelişen sektör ile beraberinde belirli sorun tiplerinde düzenli olarak aynı algoritmaların ve işleyişlerin kullanılması sebebi ile birlikte olarak yazılımların otomatik olarak gerekli algoritmalar vasıtası ile öğrenme ve tahminleme yapabilmeleri işidir. Beraberinde bu iş birçok işletmenin iş yükünü azaltmakta ve isterlerini karşılayabilmekle birlikte aynı şekilde veri bilimcilerinin yükünü de azaltmaktadır.

Optiwisdom firmasının çıkartmış olduğu optiscorer isimli yapay zeka motoru gerekli her türlü işleyişi ve skorlamayı sağlayabilmektedir.

İşletiş için yapay zeka motorunun işleyebilmesi için bir örnek olarak eğitim ve test kümeleri temin edilmelidir. Örnek alım süreci için df.sample(n) methodu kullanılarak bir örneklem alınmıştır. Beraberinde bu motorun 5000 satır limiti bulunduğundan ötürü 5000 train ve 5000 test kümeleri alınması gerekmektedir.

Alınan örnekler sonucu R2 değerimiz -0.179 olarak bulunmuştur. Kendi hiperparametreli modelimiz ise bu aşamada 0.101 olarak bulmuştur. Veri setinin dengesiz dağılımından kaynaklı olarak böyle bir olayın gerçekleşmesi istatiksel açıdan mümkündür. Beraberinde verilerimizi doğrulayabilmek adına bir eğitim ve test kümeleri yerine tek bir küme üzerinden model doğrulama yöntemlerinde sıklıkla kullanılan cross validaton yöntemi kullanılacaktır.

|  |  |
| --- | --- |
| lgbm2=LGBMRegressor(n\_estimators=600,learning\_rate=0.1)  a=cross\_val\_score(estimator=lgbm2,X=bagımsız,y=bagımlı,cv=5)  bolumlendirmek=max(a)  result = np.where(a == bolumlendirmek)  **print**("en yüksek : ",bolumlendirmek, " Ortalama: " ,a.mean())   |  | | --- | | >>>  en yüksek : 0.17266985383112998 Ortalama: 0.12522712589000573 | |

Optiscorer hız ve performasn konusunda LightGBM algoritmasını kullanması sebebi ile makale boyunca geliştirilmiş olan model ile optsicorer modeli karşılaştırılacaktır. Bu aşamada random\_state = 0 parametresi ile beraberinde elde edilen ortalama değer 0.125 olarak hesaplanmıştır. Beraberinde elde edilen dizi [0.17266985, 0.0896152 , 0.10482198, 0.14928758, 0.10974102] şeklindedir. Verilerin daha tutarlı bir şekilde karşılaştırıabilmesi ve veri setinin kendi içerisinde karıştırılması engellenmesi adına cross validaton yöntemi el ile geliştirilerek modeller karşılaştırılmıştır.

Yapılacak olan cross validaton yöntemleri için 5000 satırlık train veri seti kullanılarak CV=5 parametresi uygulanarak sırası ile lineer bir şekilde 1.,2.,3.,4.,5., 1000 boyutuna sahip veriler test edilerek geriye kalan veriler train olarak kullanılmıştır.

|  |  |
| --- | --- |
| ilk\_1000\_satırlık\_test=cross\_validaton\_testi.iloc[:1000,:]  train\_kumesi\_4000\_satırlık=cross\_validaton\_testi.iloc[1000:,:]  ilk\_1000\_satırlık\_test.to\_csv("1.Fold/Test.csv")  train\_kumesi\_4000\_satırlık.to\_csv("1.Fold/Train.csv")  sonuc=pd.read\_csv("1.Fold/LGBM\_1.Fold.csv")  **print**("Optiscorer LGBM için sonuçlar : ",r2\_score(ilk\_1000\_satırlık\_test[["yuzde\_kac\_indirim"]],sonuc[["score"]]))   |  | | --- | | >>>  Optiscorer LGBM için sonuçlar : 0.1247194999019775 | |

Yapılacak olan kod örneği üstteki gibi olup bir test ve train işleminden geçmiş bulunmaktadır. Beraberinde yapılan regreson sonucu 0.124 gibi bir değer elde etmektedir. Kendi modelimde gerçekleşen bu değer 0.188 olarak gerçekleşmiştir. Beraberinde varsayılan parametreli bir model oluşturulduğunda 0.188 olarak gerçekleştirmiştir.

2. parçada elde edilen değerler için Optsicorer kendisi 0.131 değerini elde ederken hiperparametreli modelim 0.089 olarak bir değer elde etmiştir. Taktir edilmelidir ki veri setinin düzensiz olması ve veri setinin istatiksel modeller haricinde rastgele seçilmiş olması buna bir etken olarak gösterilebilmektedir. Beraberinde varsayılan parametreli bir model oluşturulduğunda 0.136 olarak gerçekleştirmiştir.

3.parçada gerçekleşen işlemler ise optsicorer için -0.162, Hiperparametreli modelimiz için 0.101, varsayılan model için 0.155 olarak hesaplanmıştır

4. parçada ise işleyiş optsicorer için -0.162, hiperparametreli model için 0.448, varsayılan parametreli model için -0.258 olarak hesaplanmıştır

5.parçada ise işleyiş optsicorer için 0.101, varsayılan parametreli model için 0.144 ve hiperparametreli model için 0.109 olarak gerçekleşmiştir.

Cross validaton sonuçlarının 3 model üzerinden tartışmamız gerekirse

|  |  |  |  |
| --- | --- | --- | --- |
|  | Varsayılan Parametreli | Optiscorer | Hiperparametreli Model |
| İşlem 1 | 0.188 | 0.124 | 0.172 |
| İşlem 2 | 0.138 | 0.131 | 0.089 |
| İşlem 3 | 0.155 | -0.162 | 0.104 |
| İşlem 4 | -0.256 | -0.162 | -0.448 |
| İşlem 5 | 0.144 | 0.101 | 0.109 |
| Toplam: | 0.0738 | 0.0064 | 0.0052 |

Yapılan testler sonucu bazı çıkarımlarda bulunulmuştur. Hem tüm makalenin özeti olarak sayılabilecek olan beraberinde modeller ve değerlerin sorunları ile ilgili tespitler bu aşamada gerçekleştirilmiştir.

* Elde edilen büyük veriye bakıldığında elde edilen değerlerin bir çoğu modelleme için uygun olmayıp beraberinde küçük bir veri setinden elde edilen modeller büyük veriye uygulandığı zaman çok düşük doğruluklarda çıkamaktadır. Gerek demografik gerekse ürün özellikleri içerisindeki unbalance durum sebebi ile makine modelleme yapamamakta ve veriler tutarsız olması sebebi ile iyi regresyon değerleri elde edilememektedir.
* İkinci aşamada yapılan tüm veri seti doğrulanarak yapılan regresyon tipi en sağlıklı regresyon tipi olup küçük veriler üzerinde uygulandığında ortaya çıkan durum dahada iyileşmektedir. Büyük veri setlerinde ezberlemeye gitmek mantıklı yöntemlerken küçük veri setlerinde ezberlemeye gitmek tutarsızlık oluşturabilmektedir.
* Beraberinde elde edilen knime ve sklearn çalışmaları her ne kadar aynı veri setleri içerisinde modellemeler gerçekleşse ve yakın değerler elde edilse dahi tam manası ile aynı değerleri elde edememin birkaç sebebi vardır. Veride eğitime ve teste giren veriler aynı veriler olmayıp beraberinde varsayılan model parametreleri sebebi ile aralarında fark elde edilmektedir. Seçilen değişkenler farklı olup hiperparametre turninglerinde birbirleri ile aynı değerlere yaklaşılmıştır.
* Beraberinde elde edilen modeller ile ilgili olarak her ne kadar model isimleri aynı gibi gözüksede Knime ile sklearn arasındaki farklılıklar incelendiğinde gerek varsayılan model parametreleri gerekse model parametrelerinin aynı bir bicinde ayarlanamaması farklılıkların başlıca göstergesidir.
* Son aşamada optsicorer gibi bir motor ile yapılan teslterin düşük çıkması sebeplerine gelinecek olursa yapılan test ve incelemelerde elde edilen değerlerde ilk aşamada ki gibi verilerin karıştırılarak yapılan tahmin çok daha iyi bir değer iken son aşamada lineer bir bölüntüleme ile almamız regresyondaki kötülüğü arttırmaktadır.

**Kaynakça:**

* [1],[2] Big Data: Application Areas, Analytics and Security Dimension Dr. Ertuğrul AKTAN <https://dergipark.org.tr/tr/download/article-file/482194>
* [3] Big Data: A Survey Min Chen · Shiwen Mao · Yunhao Liu <http://www.ece.ubc.ca/~minchen/min_paper/BigDataSurvey2014.pdf>
* [4]networkworld:<https://www.networkworld.com/article/3325397/idc-expect-175-zettabytes-of-data-worldwide-by-2025.html>
* [5] ProTicaret e-ticaret akademisinin konferansı 2017
* [6] İşletme Ekonomi ve Yönetim Araştırmaları Dergisi Journal of Business Economics and Management Research Yıl: 2018 Sayı: 2 / 21 – 29
* [7] understanding big costumer opinion data for market-driven product deising
* [8] https://kodlayarakhayat.com/yazilim-muhendisligi/crisp-dm-metadolojisi/

**Knime platformu için**

* https://nodepit.com/

**Sklearn ve gerekli model kütüphaneleri için**

* https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html
* https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LinearRegression.html
* https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html
* https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
* https://xgboost.readthedocs.io/en/latest/python/python\_api.html
* https://lightgbm.readthedocs.io/
* https://scikit-learn.org/stable/auto\_examples/ensemble/plot\_gradient\_boosting\_regression.html

**Konu anlaşılabilirliğinin kavranabilmesi için işleyiş boyunca kullanılan Türkçe kaynaklar**

* https://www.btkakademi.gov.tr/portal/course/python-ile-makine-ogrenmesi-11800
* https://www.udemy.com/course/veribilimi/
* https://www.udemy.com/course/python-egitimi/

**Görselleştirme kütüphanaleri**

* https://seaborn.pydata.org/
* https://matplotlib.org/