

Mevsimsel İndirim Etkisi Sınıflandırıcı API

1. PROJE TANIMI

Bu çalışma, şirketimizde satış verileri üzerinden daha etkili stratejiler geliştirmek ve veri odaklı karar süreçlerini desteklemek amacıyla yürütülen çok disiplinli bir modelleme sürecinin parçasıdır. Proje kapsamında geliştirilen modeller; müşterilerin alışveriş davranışlarını anlamak, bu davranışları yönlendirmek ve çeşitli tahminlerde bulunmak üzere tasarlanmıştır. Genel hedef, satış performansını artırmak, müşteri memnuniyetini yükseltmek ve operasyonel süreçleri optimize etmektir.

Modelleme çalışmaları şu başlıklar altında yürütülmektedir:

- **Geciken Sipariş Tahmini:** Gelecek siparişlerde yaşanabilecek olası gecikmeler önceden öngörülerek, müşterilere zamanında bilgilendirme yapılması ve müşteri deneyiminin iyileştirilmesi hedeflenmektedir.
- **Stok Döngüsü Takibi:** Stokların tükenme zamanları ve ürün talep yoğunluğu tahmin edilerek, üretim ve tedarik zinciri süreçleri daha verimli hale getirilecektir.
- **Müşteri Bazlı İndirim Güncellemeleri:** Her müşterinin alışveriş alışkanlıkları analiz edilerek, kişiselleştirilmiş indirim stratejileri geliştirilecek ve bu sayede satışlara doğrudan katkı sağlanması amaçlanmaktadır.
- **Mevsimsel Ürün Bazlı İndirim Önerileri:** Alışveriş alışkanlıklarının mevsimsel değişimi göz önünde bulundurularak, çeyrek dönemler bazında ürün odaklı indirim stratejileri belirlenecektir.

Bu kapsamlı modelleme sürecinde, ekip üyeleri belirli alt başlıklarda uzmanlaşarak süreci ortak bir hedef doğrultusunda ilerletmektedir. Bu çalışmada, bize düşen görev ise; mevsimsel etkilerin indirimlerin satışlar üzerindeki etkisini nasıl şekillendirdiğini anlamak ve buna uygun bir tahmin modeli geliştirmektir. Bu doğrultuda, veri ön işleme, sınıf dengesizliğinin giderilmesi, ölçekleme ve sınıflandırma gibi adımlar izlenerek modelleme süreci gerçekleştirilmiştir.

2. AMAÇ / MOTİVASYON

Bu çalışmada, **Northwind** veritabanı kullanılarak ürün satış performanslarını analiz etmek amacıyla bir PostgreSQL sorgusu oluşturulmuştur. Sorgu, sipariş detayları (**order_details**) tablosu ile ürünler (**products**) ve kategoriler (**categories**) tablolarını birleştirerek her bir ürün için toplam satış miktarını, toplam satış gelirini ve ortalama birim fiyatını hesaplamaktadır. Ürünler, kategori bilgileriyle birlikte gruplandırılmış ve toplam satış gelirine göre azalan sırayla sıralanmıştır. Böylece, en çok gelir getiren ürünler ve kategoriler kolayca analiz edilebilir hale gelmiştir.

İlk olarak, PostgreSQL veritabanına bağlantı kurabilmek ve veri işlemesi gerçekleştirebilmek için gerekli Python kütüphanelerini yüklememiz gerekmektedir. Bu adımda **pandas**, **sqlalchemy**, ve **psycopg2** kütüphaneleri kullanılmıştır.

Veritabanına bağlanabilmek için sqlalchemy kütüphanesi kullanılarak PostgreSQL bağlantı motoru oluşturulmuştur. Aşağıda, veritabanı bağlantısı için gerekli olan bilgiler doldurarak bağlantı gerçekleştirilmiştir:

```
from sqlalchemy import create_engine
import pandas as pd

#PostgreSQL bağlantı bilgileri
DB_USERNAME = ""
DB_PASSWORD = ""
DB_HOST = "localhost"
DB_PORT = "5432"
DB_NAME = ""

#PostgreSQL bağlantı dizesi (SQLAlchemy için)
db_url = f"postgresql://{DB_USERNAME}:{DB_PASSWORD}@{DB_HOST}:{DB_PORT}/{DB_NAME}"

#SQLAlchemy engine oluştur
engine = create_engine(db_url)
```

Şekil 1. Bağlantı kurulması için `create_engine` fonksiyonunun kullanımı.

```
query = '''

SELECT
    p.product_id,
    p.product_name,
    p.category_id,
    c.category_name,
    SUM(od.quantity) AS total_quantity,
    SUM(od.unit_price * od.quantity * (1 - od.discount)) AS total_sales,
    AVG(od.unit_price) AS avg_price
FROM
    order_details od
LEFT JOIN
    products p ON od.product_id = p.product_id
LEFT JOIN
    categories c ON p.category_id = c.category_id
GROUP BY
    p.product_id, p.product_name, p.category_id, c.category_name
ORDER BY
    total_sales DESC;'''

#Sorgu sonucu dataframe olarak alırız:
df = pd.read_sql(query, engine)
```

Şekil 2. Oluşturduğumuz SQL sorgusunu `pandas` ve `SQLAlchemy` ile çalıştırarak veri; bir `pandas` DataFrame olarak kullanılmıştır.

3. VERİ ÖN İŞLEME

Özellik Seçimi, Dengesizlik Giderme ve Ölçekleme

Veri oluşumunun ardından genel bilgilendirme almak için `df.info()` fonksiyonu kullanılmıştır. Bu fonksiyon, veri kümesindeki sütunları, veri tiplerini ve her sütunda kaç tane eksik veri bulunduğunu göstermiştir. Ayrıca, eksik verilerin toplam sayısını görmek için `df.isna().sum()` fonksiyonu kullanılmıştır. Sonuçlarımıza göre na veya null değerler olmadığından sonraki aşama olan sayısal değişkenlerin özet istatistikleri `df.describe()` fonksiyonu ile elde edilmiştir. Elde ettiğimiz tabloda aykırı değerlerin oranca fazla olması üzerine, aykırı değerlerin istatistiksel analizler üzerindeki etkisini azaltarak daha güvenilir tahminler elde edilmesini sağlamak için [Winsorize](#) metodu uygulanmıştır.

Winsorize yöntemi ile aykırı değerlerin etkisini azaltarak veriyi daha sağlıklı hale getirdikten sonra, sayısal değişkenlerin dağılımlarını incelemek amacıyla çeşitli görselleştirmeler oluşturulmuştur. İlk olarak, `unit_price` değişkeninin dağılımını anlamak için histogram ve kutu grafikleri kullanılmıştır. Histogram grafiği ile fiyat dağılımı görselleştirilirken, kutu grafiği sayesinde aykırı değerlerin varlığı daha net bir şekilde gözlemlenmiştir. Buna ek olarak, `unit_price` ile `quantity` arasındaki ilişkinin belirlenmesi için bir dağılım grafiği oluşturulmuş ve bu iki değişken arasındaki potansiyel korelasyon incelenmiştir.

4. MODEL GELİŞTİRME / TEST


Değişken değerlerin saptanması ve iyileştirmelerin yapılmasının ardından kullanılabilir formata getirilen datada; makine öğrenmesi modelleri üzerine çalışılmıştır. Makine öğrenmesi modelleri için sağlıklı ve anlamlı sonuçlar elde edebilmek adına, verilerin modele uygun hale getirilmesi oldukça önemlidir. Bu kapsamda ilk adım olarak, veri kümesindeki **bağımlı (hedef) ve bağımsız (özellik) değişkenler** ayrılmıştır. Bağımsız değişkenler **X**, modelin öğrenme sürecinde kullanacağı tüm sütunları temsil eder ve bu projede, `discount_effective` haricindeki tüm sütunları kapsamaktadır. Bağımlı değişken **y** ise modelin tahmin etmeye çalıştığı hedef sütundur ve burada `discount_effective` olarak belirlenmiştir.

Veri kümesinde sınıflar arasında **dengesizlik** bulunduğundan dolayı, modelin azınlık sınıfı doğru bir şekilde öğrenebilmesi için **SMOTE (Synthetic Minority Over-sampling Technique)** yöntemi uygulanmıştır. SMOTE, azınlık sınıfına ait örneklerin sayısını artırmak için sentetik veriler üretir. Bu yöntem sayesinde, veri kümesindeki sınıflar dengelenmiş ve modelin her iki sınıfı da adil bir şekilde öğrenmesi sağlanmıştır. Bu, özellikle dengesiz sınıf dağılımlarında modelin sadece çoğunluk sınıfı odaklanmasını engellemek açısından oldukça kritik bir adımdır.


SMOTE sonrası, modelin farklı özelliklerdeki verilerden etkilenmemesi ve öğrenme sürecinin daha verimli hale gelmesi için **ölçekleme (scaling)** işlemi uygulanmıştır. Bu amaçla, tüm sayısal veriler **Min-Max Scaling** yöntemi ile 0 ile 1 aralığına dönüştürülmüştür. Min-Max ölçekleme, her bir özelliği kendi minimum ve maksimum değerleri arasında yeniden ölçeklendirerek modelin bazı özelliklere gereğinden fazla ağırlık vermesini önlemektedir. Böylece modelin eğitimi daha dengeli ve kararlı bir hale getirilmiştir.


Modelleme sürecinde, elde edilen veriler üzerinde uygulanan analizler doğrultusunda farklı makine öğrenmesi algoritmaları denenmiş ve performansları karşılaştırılmıştır. Bu kapsamda, Linear Regression, K-Nearest Neighbors (KNN) ve Decision Tree algoritmaları kullanılarak sınıflandırma modelleri oluşturulmuştur. Her bir modelin çıktısı değerlendirilerek, mevsimsel indirimlerin satışlar üzerindeki etkisini en doğru şekilde yansıtan yapı tespit edilmeye çalışılmıştır.

Model başarılarının karşılaştırılmasında, sadece genel doğruluk oranı değil, aynı zamanda sınıflandırma kalitesini daha derinlemesine analiz edebilmek için çeşitli performans metriklerinden yararlanılmıştır:

TRUE - POSITIVE |  | : Gerçek pozitif olanı doğru pozitif tahmin etme.

TRUE - NEGATIVE |  | : Gerçek negatif olanı doğru negatif tahmin etme.

FALSE - POSITIVE |  | : Gerçek negatif olanı yanlışlıkla pozitif tahmin etme (Type I error)

FALSE - NEGATIVE |  | : Gerçek pozitif olanı yanlışlıkla negatif tahmin etme (Type II error)

- **Accuracy (Doğruluk):** Tüm sınıflandırmaların dengeli dağılım gösterdiği durumlarda başarı tespiti için kullanılabilir.

Doğru Tahmin Sayısı / Toplam Tahmin Sayısı

- **Precision (Kesinlik):** Modelin *pozitif tahminlerinden* kaç tanesinin gerçekten pozitif olduğunu ölçmek için kullanılmaktadır. Özellikle yanlış pozitiflerin maliyetli olduğu durumlarda önemlidir.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

- **Recall (Duyarlılık):** Gerçek pozitif örneklerin ne kadarını doğru tahmin ettiğimizi göstermekte kullanılmaktadır. Kaçırılmaması gereken kritik durumlarda önceliğe sahiptir.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

- **F1-Score:** Precision ve Recall'un harmonik ortalamasıdır. Bu iki metriği dengeleyerek, özellikle dengesiz veri setlerinde iyi bir değerlendirme sağlamaktadır.

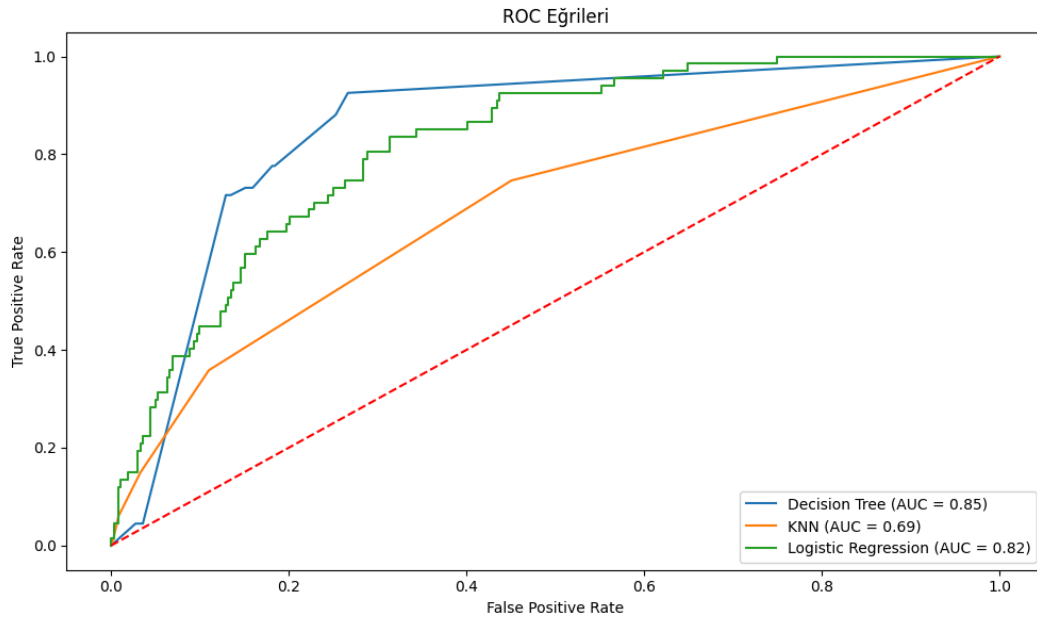
$2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$

- **ROC-AUC:** Modelin pozitif ve negatif sınıfları ayırma başarısını ölçmede kullanılmaktadır.

Skor 1'e yaklaştıkça; modelin ayırt etme gücü artmaktadır. 1.0 = en iyi ayırım, 0.5 = düşük başarılı ayırım

Model performansının değerlendirilmesi için test veri seti üzerinde yapılan analizler sonucunda, seçilen Decision Tree algoritmasının sınıflandırma başarısı dikkat çekici düzeydedir. Modelin doğruluk (accuracy) oranı %85 olarak ölçülürken, F1-Score, Precision ve Recall gibi metriklerde de dengeli ve başarılı sonuçlar elde edilmiştir.

Modelin ROC-AUC skoru, sınıflar arasında ayırt etme gücünün yüksek olduğunu göstermekte ve modelin genellenebilirliğini desteklemektedir. Aşağıdaki grafiklerde confusion matrix ve ROC eğrisi görselleştirilerek, modelin tahmin performansı detaylandırılmıştır.



Şekil 3. Model performans metriklerinin tablosu ve ROC eğrisi.

En iyi performansı gösteren Decision Tree (Karar Ağacı) modeli, nihai tahminleme sürecinde kullanılmıştır. Karar Ağaçları; sınıflandırma problemlerinde sıkça tercih edilen, veriyi belirli karar kurallarına göre dallara ayırarak tahmin yapan sezgisel ve güçlü bir algoritmadır. Modelin her bir dalı, veri üzerinde alınan bir kararı temsil ederken, yaprak düğümler ise sınıflandırma sonuçlarını oluşturur.

Modelin genel başarımını artırmak ve aşırı öğrenme (overfitting) gibi olumsuz durumları önlemek amacıyla bazı hiperparametre ayarlamaları yapılmıştır. Bu ayarlamalar, modelin hem esnekliğini hem de genellenebilirliğini kontrol etmek için kritik öneme sahiptir:

max_depth: Ağacın maksimum derinliğini belirler. Bu parametre, ağacın ne kadar büyüyeceğini ve dolayısıyla modelin ne kadar karmaşık olacağını kontrol eder. Aşırı büyük değerler overfitting'e yol açabilir.

min_samples_split: Bir iç düğümün daha küçük düğümlere ayrılabilmesi için gerekli minimum örnek sayısını belirler. Bu parametre ile ağacın fazla bölünmesini engelleyerek daha dengeli modeller üretilebilir.

min_samples_leaf: Her yaprak düğümde bulunması gereken minimum örnek sayısıdır. Bu parametre ile çok küçük ve anlamsız yapraklar oluşması engellenir.

Bu hiperparametrelerin dikkatli bir şekilde ayarlanması, modelin hem eğitim verisine uyumunu hem de yeni veriler üzerindeki tahmin gücünü optimize etmiştir.

5. SONUÇ / MODELİN SATIŞ STRATEJİSİNE KATKISI

Geliştirilen mevsimsel sınıflandırma modeli, pazarlama ve satış ekiplerinin çeyrek dönemler bazında daha isabetli indirim stratejileri geliştirmesine yardımcı olacaktır. Ürün bazında hangi dönemlerde indirim yapılmasının daha etkili olduğu öngörülebilecek; bu sayede hem stok yönetimi daha verimli hale gelecek hem de müşteri davranışları doğru yönlendirilebilecektir.

Ayrıca bu model sayesinde, sadece geçmiş verilere bağlı kalmadan, mevsimsel eğilimlere dayalı proaktif stratejiler oluşturulabilecek ve kampanyaların dönüşüm oranları artırılabilir.

Modelin FastAPI ile servis haline getirilmesi sayesinde, bu öngörüler sadece analiz düzeyinde kalmayıp, gerçek zamanlı şekilde sistemlere entegre edilebilmektedir. Satış ekipleri, canlı verilerle API üzerinden tahmin alarak, anlık olarak hangi ürünlerde indirim yapılmasının etkili olacağını öğrenebilmekte; bu sayede dinamik kampanya yönetimi sağlanmaktadır.

Bu yapı, özellikle hızlı karar alma süreçlerinin kritik olduğu dönemsel kampanyalar ve stok yönetimi açısından şirket için büyük bir operasyonel avantaj sunmaktadır.