Makine Öğrenmesi – Dönem Ödevi Projesi (F2025)

Hazırlayan: Ahmet Furkan TOPRAK

Öğrenci No: 202213709044

Sınıflandırma

1. Veri Setinin Tanıtımı

Bu çalışmada kullandığım veri setinin adı **“Adult”** (veya literatürde bilinen diğer adıyla **“Census Income”**) veri setidir. Veri seti, **“UC Irvine Machine Learning Repository”** adlı siteden alınmıştır ve orijinal kaynağa ait erişim linki şudur: *https://archive.ics.uci.edu/dataset/2/adult*.

Bu veri setinden yola çıkılarak amaç; 1994 yılında yapılan nüfus sayımı verilerine dayanarak bireylerin yaş, eğitim, meslek ve medeni hal gibi özelliklerini analiz etmek ve yıllık gelirlerinin 50000$’ın üzerinde olup olmadığını tahmin ederek sınıflandırmaktır.

Veri setinde toplam 48842 örneklem ve 14 adet öznitelik (feature) bulunmaktadır. Bu yönleriyle ödev için belirtilen en az 10000 örneklem ve 10 öznitelik şartını karşılamaktadır. 14 öznitelik şu şekilde açıklanabilir:

- **Hedef Değişken:** ‘income’-> Bireylerin gelir düzeyini ifade eder. <=50K veya >50K olmak üzere iki sınıflıdır.
- **Kategorik Değişkenler:** ‘workclass’, ‘education’, ‘marital-status’, ‘occupation’, ‘relationship’, ‘race’, ‘sex’, ‘native-country’ niteliklerini içermektedir.
- **Sayısal Değişkenler:** ‘age’, ‘fnlwgt (final weight)’, ‘education-num’, ‘capital-gain’, ‘capital-loss’, ‘hours-per-week’ değerlerini içermektedir.

Verilerin tümü, veri setinin internet sitesinde belirtilen Python import’u ile çekilmiştir. Bunun için **ucimlrepo** kütüphanesi kullanılmıştır. Veri çekildikten sonra Data Frame’e dönüştürülmüş, bütünlüğü görmek ve eksik ya da hatalı bilgileri bulmak amacıyla head(), info(), describe(), isnull().sum() fonksiyonlarına başvurulmuştur. Bu fonksiyonların çıktıkları aşağıdaki gibidir:

```
...  0   39          State-gov   77516  Bachelors             13
     1   50   Self-emp-not-inc   83311  Bachelors             13
     2   38            Private  215646    HS-grad              9
     3   53            Private  234721       11th              7
     4   28            Private  338409  Bachelors             13

          marital-status         occupation   relationship   race     sex
     0        Never-married       Adm-clerical  Not-in-family  White    Male
     1   Married-civ-spouse    Exec-managerial        Husband  White    Male
     2             Divorced  Handlers-cleaners  Not-in-family  White    Male
     3   Married-civ-spouse  Handlers-cleaners        Husband  Black    Male
     4   Married-civ-spouse     Prof-specialty           Wife  Black  Female

        capital-gain  capital-loss  hours-per-week native-country income
     0          2174             0              40  United-States  <=50K
     1             0             0              13  United-States  <=50K
     2             0             0              40  United-States  <=50K
     3             0             0              40  United-States  <=50K
     4             0             0              40           Cuba  <=50K
     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 48842 entries, 0 to 48841
     Data columns (total 15 columns):
      #   Column          Non-Null Count  Dtype
     ---  ------          --------------  -----
      0   age             48842 non-null  int64
      1   workclass       47879 non-null  object
      2   fnlwgt          48842 non-null  int64
      3   education       48842 non-null  object
      4   education-num   48842 non-null  int64
      5   marital-status  48842 non-null  object
      6   occupation      47876 non-null  object
      7   relationship    48842 non-null  object
      8   race            48842 non-null  object
      9   sex             48842 non-null  object
      10  capital-gain    48842 non-null  int64
      11  capital-loss    48842 non-null  int64
      12  hours-per-week  48842 non-null  int64
      13  native-country  48568 non-null  object
      14  income          48842 non-null  object
     dtypes: int64(6), object(9)
     memory usage: 5.6+ MB
     None
```

```
                    age        fnlwgt  education-num  capital-gain  capital-loss  \
     count  48842.000000  4.884200e+04   48842.000000  48842.000000  48842.000000
     mean      38.643585  1.896641e+05      10.078089   1079.067626     87.502314
     std       13.710510  1.056040e+05       2.570973   7452.019058    403.004552
     min       17.000000  1.228500e+04       1.000000      0.000000      0.000000
     25%       28.000000  1.175505e+05       9.000000      0.000000      0.000000
     50%       37.000000  1.781445e+05      10.000000      0.000000      0.000000
     75%       48.000000  2.376420e+05      12.000000      0.000000      0.000000
     max       90.000000  1.490400e+06      16.000000  99999.000000   4356.000000

           hours-per-week
     count    48842.000000
     mean        40.422382
     std         12.391444
     min          1.000000
     25%         40.000000
     50%         40.000000
     75%         45.000000
     max         99.000000
     age                 0
     workclass         963
     fnlwgt              0
     education           0
     education-num       0
     marital-status      0
     occupation        966
     relationship        0
     race                0
     sex                 0
     capital-gain        0
     capital-loss        0
     hours-per-week      0
     native-country    274
     income              0
     dtype: int64
```

Bu çıktılardan yola çıkarak yapılan veri ön işleme adımları bir sonraki bölümde anlatılmaktadır.

2. Uygulanan Yöntemler

2.1. Çalışma Ortamı ve Donanım Bilgileri

Proje, Google Colab üzerinden Python programlama dili kullanılarak geliştirilmiştir. Kod dosyaları .ipynb olarak kaydedilmiştir. Kullanılan diğer özellikler şöyledir:

- **GPU (Grafik İşlem Birimi):** NVIDIA Tesla T4 (16 GB VRAM, 2560 CUDA çekirdeği)
- **RAM (Sistem Belleği):** ~12.7 GB kullanılabilir sistem belleği.
- **İşlemci (CPU):** Intel Xeon @ 2.20GHz (Çift çekirdekli sanal işlemci).

Karmaşık algoritmaların eğitimi ve GridSearchCV ile yapılan hiperparametrelerin optimizasyonu sürecinde CPU yerine T4 GPU kullanımıyla eğitim süresi önemli ölçüde düşmüştür. Ayrıca matris işlemlerinin paralelleştirilmesi sayesinde işlem maliyeti minimize edilmiştir.

2.2. Veri Ön İşleme (Data Preprocessing)

Ham veri seti üstünde model başarısını arttırmak ve algoritma gereksinimlerini karşılamak için aşağıdaki yöntemlere başvurulmuştur:

2.2.1. Eksik Veri Analizi

- Veri setinde bazı boş değerler, teknik olarak NaN yerine "?" karakteri ile kodlanmıştır. **df_clean** isimli yeni bir Data Frame oluşturarak buradaki bu karakterler NaN formatına dönüştürülmüştür.
- Bu dönüşüm sonucunda df_clean Data Frame'i için tekrardan isnull().sum() fonksiyonu çağırılarak hangi özniteliklerden ne kadar boş değer olduğu tespit edilmiştir. Tespit sonucunda **workclass için 2799**, **occupation için 2809** ve **native-country için 857** NaN veri olduğu anlaşılmıştır.
- workclass, occupation ve native-country sütunlarında tespit edilen eksik veriler, veri kaybını önlemek adına ilgili sütunun **Mod** değeri ile doldurulmuştur.
- Bu işlemlerin ardından df_clean için isnull().sum() fonksiyonu bir kez daha çağırılmış ve herhangi boş bir değere rastlanmamıştır.

2.2.2. Öznitelik Kodlama

Makine Öğrenmesi algoritmaları, matematiksel işlemler yaptığı için kategorik olan öznitelikler object veri tipinden sayısal veri tipine dönüştürülmüştür. Bunun için Label Encoding ve One-Hot Encoding yöntemlerinin her ikisi de kullanılmıştır.

- **Label Encoding:**

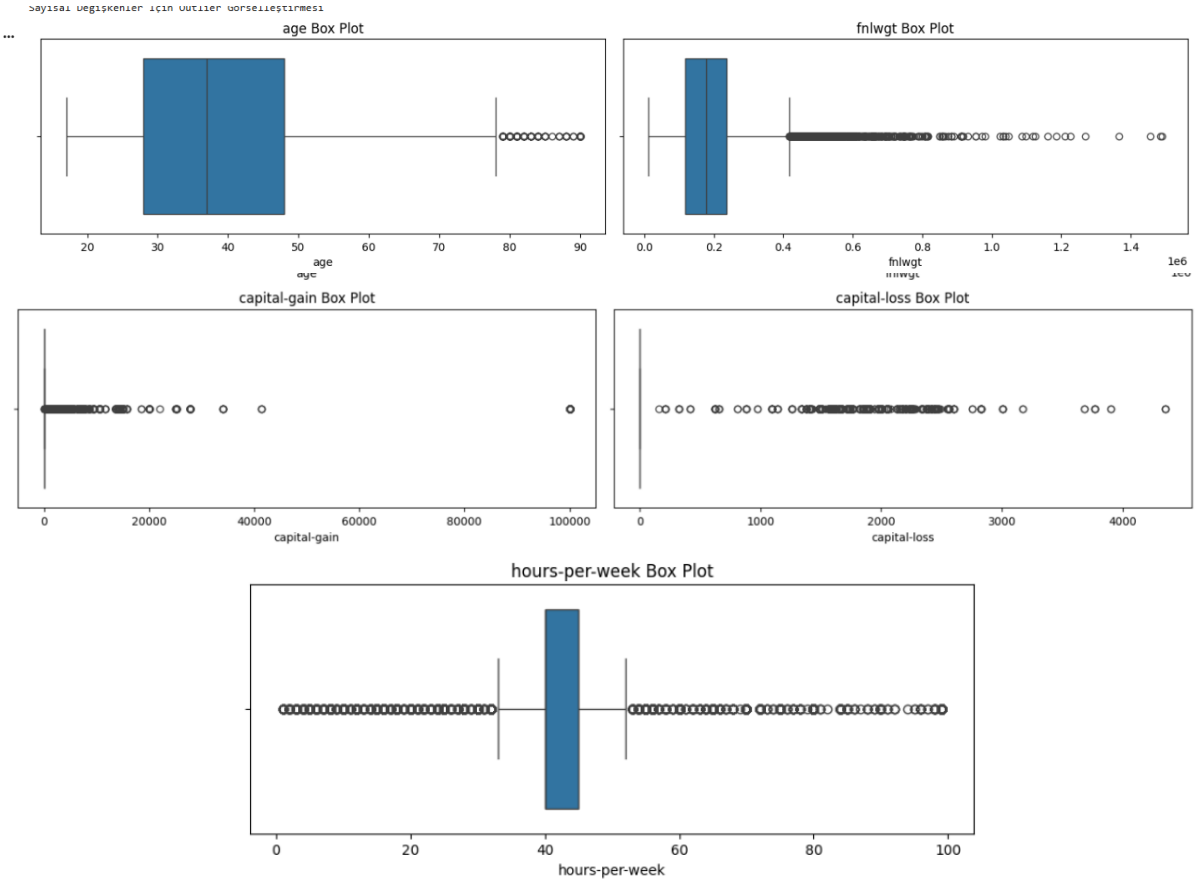
İlk olarak, hedef değişken olarak belirlenen **income** değişkenine Label Encoding yöntemiyle 0-1 dönüşümü yapılmıştır. Bu işlemde, **<=50K** değerleri için 0, **>50K** değerleri için ise 1 kodlaması uygulanmıştır. Başlangıçta map fonksiyonuna sadece **<=50K** ve **>50K** değerleri verilmiş, ancak bazı verilerin tam dönüşmediği görülmüştür. Bunun üzerine **income** değişkeninin unique değerleri incelenmiştir. Hatanın, bazı verilerin **<=50K.** ve **>50K.** şeklinde (sonunda nokta ile) yazılmasından kaynaklandığı anlaşılmıştır. Sonuç olarak bu durum için bir sözlük oluşturulmuş ve tanımlı olan 4 veri türü için de 0 ve 1 dönüşümleri kodlanmıştır.

- **One-Hot Encoding:**

Sıralama içermeyen tüm kategorik değişkenler (daha önce veri seti tanıtımında belirtilen), pd.get_dummies fonksiyonu yardımıyla her kategori yeni öznitelik olacak şekilde eklenmiştir.

2.2.3. Aykırı Değer (Outlier) Temizliği

Veri setindeki sayısal değişkenlerin dağılımını incelemek ve aykırı değerleri tespit etmek amacıyla **Boxplot** analizi uygulanmıştır. Bu analiz sayesinde verilerin öznitelikler üzerindeki dağılımı görselleştirilmiş ve değerlerin gerçekçi olup olmadığı değerlendirilmiştir.

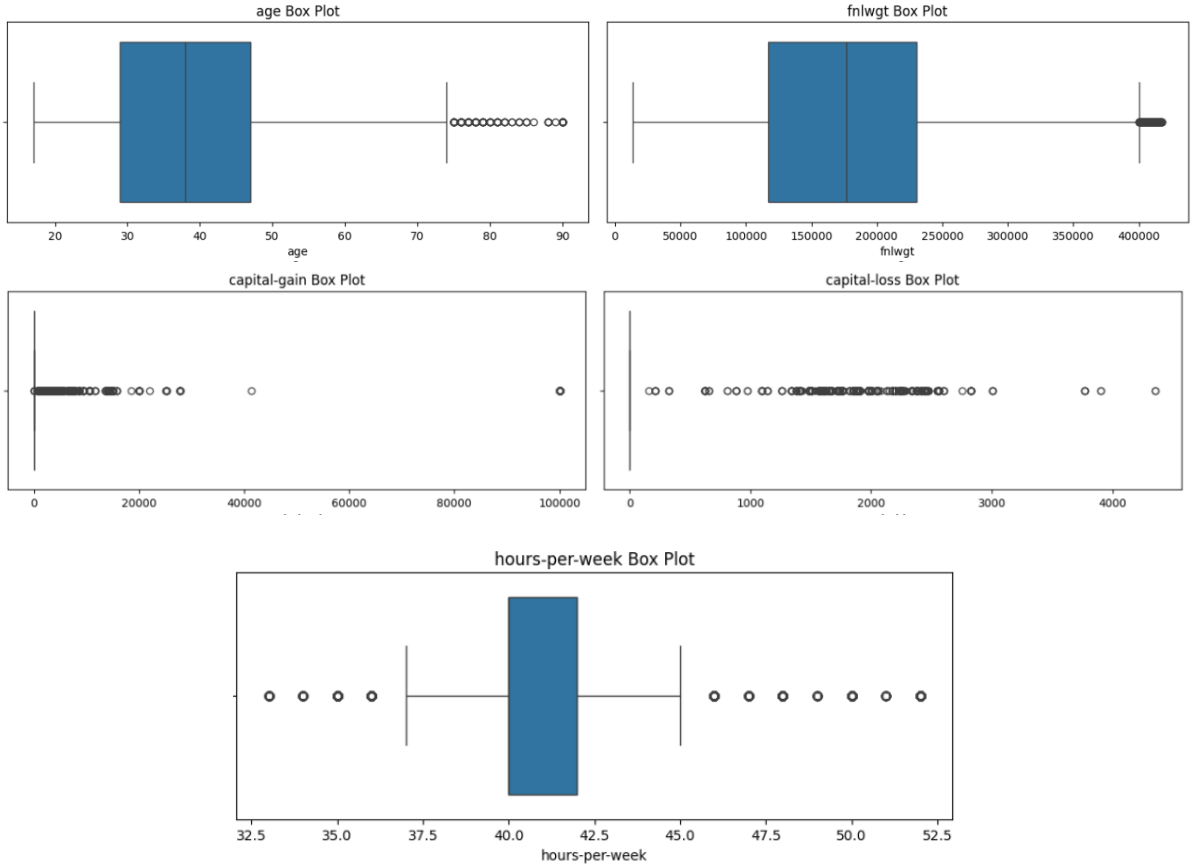


Grafikler üzerindeki gözlemler ve alınan kararlar şöyledir:

- **age:** Verilerin büyük çoğunluğunun 25-50 yaş aralığında yoğunlaştığı görülmüştür. 80-90 yaş bandındaki değerler uç noktalarda yer alsa da gerçeklik açısından olağan kabul edilmiştir.
- **fnlwgt:** Veriler genel olarak 0.2 seviyesinde yoğunlaşmıştır. Ancak bu yoğunluğun 7 katına kadar çıkan değerler gözlemlenmiş olup bu durum belirgin bir aykırılığa işaret etmektedir.
- **capital-gain ve capital-loss:** Bu değişkenlerde, standart bir kutu dağılımı gözlemlenememiştir. Veriler, sıfır noktasında yığılmış olsa da çok yüksek değerler de mevcuttur. Hedef değişken olan 'gelir düzeyi' ile doğrudan ilişkili oldukları (sermaye kazancı/kaybı geliri belirlediği) için bu yüksek değerler aykırı değer olarak nitelendirilmemiş ve veri setinde korunmuştur.

- **hours-per-week:** Verilerin medyan değeri beklendiği üzere 40 saat civarındadır. Ancak haftalık 100 saati bulan çalışma süreleri tespit edilmiştir. İnsan fizyolojisi ve çalışma standartları göz önüne alındığında, bir kişinin sürekli olarak haftada 100 saat çalışması gerçekçi görünmediğinden bu veriler aykırı değer olarak değerlendirilmiştir.

Yapılan değerlendirmeler sonucunda, fnlwgt ve hours-per-week özniteliklerindeki tutarsızlıkları gidermek için **IQR (Interquartile Range)** yöntemi kullanılmıştır. Q1 (%25) ve Q3 (%75) çeyreklikler hesaplanarak (1.5 x IQR) sınırının dışında kalan gözlemler veri setinden çıkarılmıştır. Bu temizleme işlemi ile veri setindeki gürültü azaltılmış ve modelin genelleme yeteneğinin artırılması hedeflenmiştir. Bu işlemler yapıldıktan sonra boxplot grafikleri tekrardan çizilmiş ve aykırılıkların azaldığı gözlemlenmiştir.



Veri setinde tespit edilen aykırı değerler ve çıkarma işlemi sonrası veri setinin durumu aşağıdaki gibidir:

```
...     -> fnlwgt: 1453 adet aykırı değer tespit edildi.
        -> hours-per-week: 13496 adet aykırı değer tespit edildi.
----------------------------------------
Önceki Veri Sayısı : 48842
Silinen Veri Sayısı: 14590
Yeni Veri Sayısı   : 34252
Veri Kaybı Oranı   : %29.87
```

2.2.4. Özellik Ölçekleme (Feature Scaling):

Özellikle KNN gibi mesafe temelli algoritmaların performansını etkilememesi için sayısal veriler; **StandardScaler** kullanılarak ortalaması 0, standart sapması 1 olacak şekilde

ölçeklenmiştir. Bu işlem, data leakage'ı önlemek adına modelin train ve test olarak ayrılmasından sonra yapılmıştır.

2.3. Model Eğitimi ve Değerlendirme Stratejisi

Veri ön işleme adımlarından geçen df_clean'i kullanarak hedef değişken olan income özniteliği drop() fonksiyonu yardımıyla düşürülmüş ve X isimli değişkenine atanmıştır. y değişkenine ise income özniteliği atanmıştır. Veri seti, sınıf dağılım dengesini korumak amacıyla stratify=y parametresi kullanılarak **%80 Train** ve **%20 Test** seti olarak ayrılmıştır.

Çalışmada üç temel algoritma kullanılmıştır:

- **K-Nearest Neighbors (KNN):** Mesafe temelli, parametrik olmayan temel bir karşılaştırma modeli olarak seçilmiştir.
- **Random Forest:** Aşırı öğrenmeyi (overfitting) engellemek ve özellik önemini belirlemek için Bagging yöntemini kullanan bir topluluk modelidir.
- **XGBoost (Extreme Gradient Boosting):** Yüksek performansı ve GPU desteği nedeniyle tercih edilen, Gradient Boosting tabanlı gelişmiş bir modeldir.

Her model için **5-Katlı Çapraz Doğrulama (5-Fold Cross Validation)** ile entegre edilmiş **GridSearchCV** yöntemi kullanılarak en iyi hiperparametreler (neighbors sayısı, ağaç derinliği, öğrenme oranı vb.) tespit edilmiştir.

3. Sonuçlar ve Metrikler

3.1. Özellik Seçimi (Feature Selection) ve Optimizasyon

Çalışmada veri boyutunu yönetilebilir tutmak ve modelin gürültüden (noise) arındırılması için iki aşamalı özellik seçimi uygulanmıştır:

3.1.1. Sıfır Varyans Analizi

Label Encoding ve One-Hot Encoding sonrası, tüm örnekler için sabit değer alan ve ayırt ediciliği olmayan sütunlar kontrol edilmiştir. (Bu çalışmada sıfır varyanslı sütun bulunamamıştır, ancak kontrol adımı uygulanmıştır.)

3.1.2. Permutation Importance (Önem Derecesine Göre Eleme)

Her model (KNN, RF, XGBoost) için "Permutation Importance" yöntemi ile özelliklerin modele katkısı hesaplanmıştır. Katkısı olmayan veya negatif etki yapan (Importance <= 0) öznitelikler veri setinden çıkarılarak modeller optimize edilmiş veri setiyle tekrar eğitilmiştir.

- **KNN:** 61 gereksiz öznitelik çıkarıldı, 36 öznitelik ile devam edildi.
- **Random Forest:** 67 öznitelik çıkarıldı, 30 öznitelik ile devam edildi.
- **XGBoost:** 49 öznitelik çıkarıldı, 48 öznitelik ile devam edildi.

3.1.3. GridSearchCV

Makine öğrenmesi modellerinin başarısı, eğitim öncesinde belirlenen ve modelin öğrenme yapısını kontrol eden "hiperparametre" değerlerine doğrudan bağlıdır. Rastgele seçilen

parametrelerin modelin performansını (accuracy, f1-score) düşürme riski taşıması nedeniyle bu çalışmada **GridSearchCV (Izgara Araması)** yöntemi kullanılmıştır.

Manuel deneme-yanılma yönteminin aksine, GridSearchCV belirlenen parametre uzayındaki tüm olası kombinasyonları sistematik olarak dener. Bu sayede:

- Belirlenen aralıktaki global optimum değerlerin gözden kaçırılması engellenmiştir.
- Modelin sadece eğitim verisinde değil, genel yapıda da başarılı olması için optimizasyon süreci **5-Katlı Çapraz Doğrulama** ile entegre edilmiştir.
- Her model için kritik öneme sahip parametreler belirlenmiş ve aşağıdaki gibi taranmıştır:

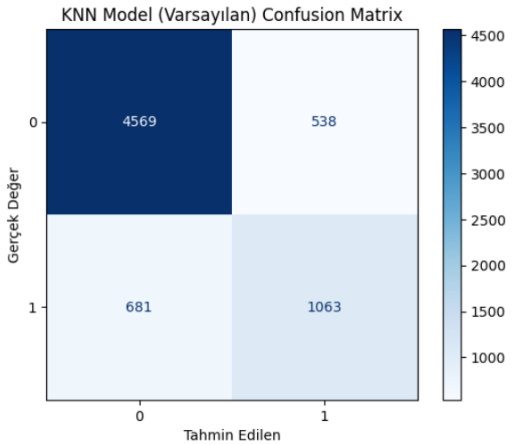
1. **KNN:** En yakın komşu sayısı (*n_neighbors*) ve mesafe metriği (*metric*) optimize edilmiştir.
2. **Random Forest:** Karar ağacı sayısı (*n_estimators*), maksimum derinlik (*max_depth*) ve bölünme kriteri (*criterion*) taranmıştır.
3. **XGBoost:** Öğrenme oranı (*learning_rate*), ağaç derinliği (*max_depth*) ve örneklem oranı (*subsample*) gibi parametreler optimize edilmiştir.

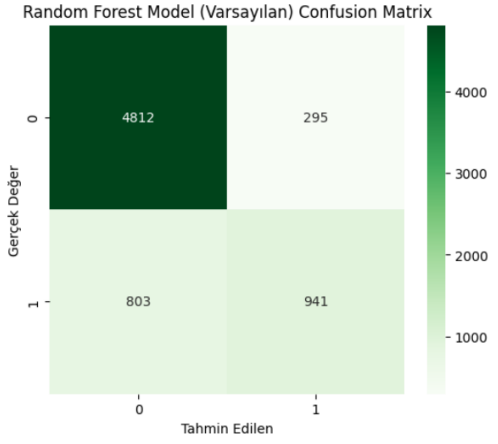
3.2. Algoritma Bazlı Eğitim Süreçleri

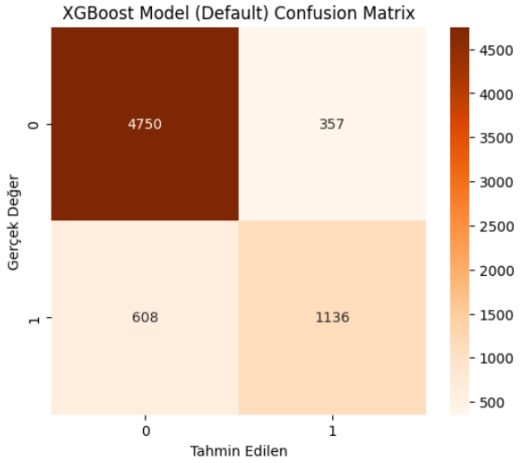
Bu çalışmada kullanılan her üç model için sistematik olarak **Eğitim – Öznitelik Seçimi – Optimizasyon'** döngüsü uygulanmıştır. Modellerin eğitimi sırasında aşağıdaki adımlar izlenmiştir. Ayrıca KNN ve Random Forest için **cuML** kütüphanesinin GPU hızlandırmalı versiyonu, XGBoost için de parametre ayarı ile GPU kullanılarak eğitim süreleri kısaltılmıştır.

3.2.1. Varsayılan (Default) KNN, Varsayılan (Default) Random Forest, Varsayılan (Default) XGBoost

Modellerin temel performansını gözlemlemek amacıyla bütün hiperparametreler kütüphane dokümantasyonlarında**[1,3,4]** belirtilen default halleriyle kullanılmıştır. Ayrıca bu modellerde, tüm öznitelikler kullanılmış ve eğitim sonucunda modellerin verdiği önem düzeyleri grafik olarak ortaya konmuştur. Modelin eğitiminin ardından test setiyle model test edilmiş ve karmaşıklık matrisi (confussion matrix) çıktısı alınmıştır. Aşağıda modeller için karmaşıklık matrisi çıktıları ve modellerin önem katsayı grafikleri bulunmaktadır.







KNN için Önem Katsayıları Grafiği

Random Forest için Önem Katsayıları Grafiği

XGBoost için Önem Katsayıları Grafiği

3.2.2. Optimize Edilmiş KNN, Optimize Edilmiş Random Forest, Optimize Edilmiş XGBoost

Bütün modellerde, default değerlerde önem sayısı az olan (bir önceki bölümde anlatılan **Permutation Importance** kullanılarak) öznitelikler çıkarılmış ve GridSearchCV (yine bir önceki bölümde anlatılmıştır.) yardımıyla hiperparametre optimizasyonu yapılmıştır. Random Forest ve KNN; cuML kütüphanesi, XGBoost ise kendi kütüphanesi ile eğitildiği için hiperparametrelerin belirlenmesinde yine kütüphane dokümantasyonlarından**[2,4]** yararlanılmıştır. En iyi parametreler **Accuracy** yardımıyla bulunmuştur. Seçilen en iyi parametreler, en iyi cross-validation skoru ve karmaşıklık matrisi tüm modeller için aşağıda verilmiştir.

KNN Optimizasyonu için:

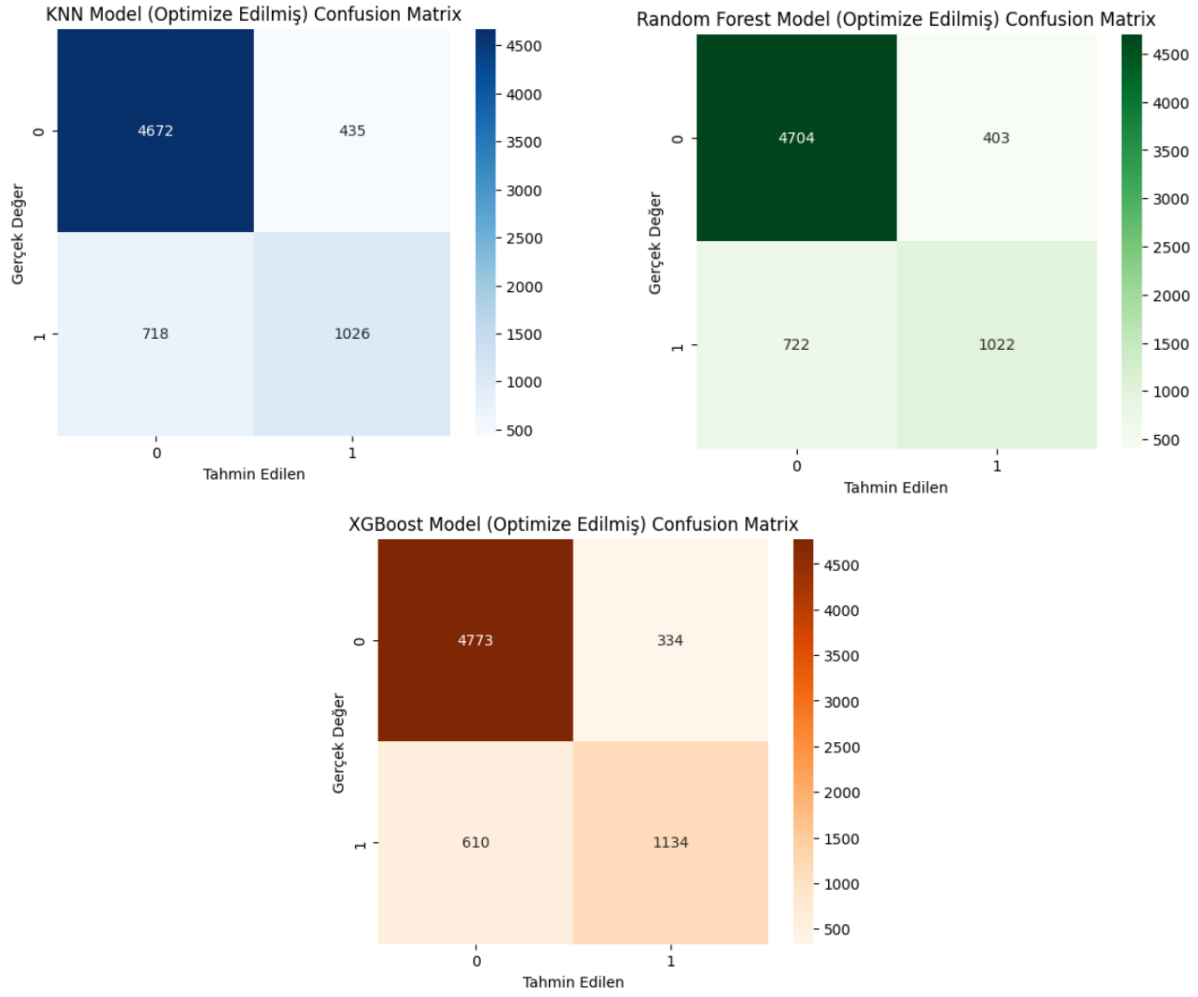
```
En İyi Parametreler: {'metric': 'manhattan', 'n_neighbors': 21, 'weights': 'uniform'}
En İyi Cross-Validation Skoru: 0.8324
```

Random Forest Optimizasyonu için:

```
En İyi Parametreler: {'max_depth': 15, 'max_features': 'log2', 'n_estimators': 100, 'split_criterion': 0}
En İyi Cross-Validation Skoru: 0.8408
```

XGBoost Optimizasyonu için:

```
En İyi Parametreler: {'learning_rate': 0.3, 'max_depth': 3, 'n_estimators': 300, 'subsample': 1.0}
En İyi Cross-Validation Skoru: 0.8637
```



3.3. Model Performansının Ölçülmesi İçin Kullanılan Metrikler

Bu çalışmada, geliştirilen sınıflandırma modellerinin başarımını ölçmek ve karşılaştırmak amacıyla literatürde kabul görmüş beş temel metrik kullanılmıştır. Veri setindeki sınıf dağılımı dengesiz olduğundan (Geliri <=50K olanların sayısı daha fazla) ve modelin sadece doğruluk oranına bakmak yanıltıcı olabileceğinden F1-Score ve ROC-AUC gibi daha hassas metriklere öncelik verilmiştir.

3.3.1. Accuracy (Doğruluk)

- **Formül:** (TP+TN) / (TP+TN+FP+FN)

- Genel bir başarım göstergesi olarak kullanılmıştır. Ancak veri setindeki dengesizlik nedeniyle modelin çoğunluk sınıfını (<=50K) tahmin ederek yüksek doğruluk elde etme riski olduğundan tek başına bir başarı kriteri olarak değerlendirilmemiştir.

3.3.2. Precision (Kesinlik)

- **Formül:** TP / (TP+FP)
- Modelin yanlış alarm verme oranını (False Positive) düşük tutup tutmadığını anlamak için kullanılmıştır. Yani, "Geliri yüksek dediğimiz kişilerin gerçekten geliri yüksek mi?" sorusuna yanıt aranmıştır.

3.3.3. Recall (Duyarlılık)

- **Formül:** TP / (TP+FN)
- Modelin gözden kaçırdığı yüksek gelirli bireyleri tespit etmek için kritiktir. Düşük recall değeri, modelin yüksek gelirli kişileri tespit etmekte zorlandığını gösterir.

3.3.4. F1-Score

- **Formül:** 2*(Precision*Recall) / (Precision+Recall)
- Sınıf dengesizliği olan veri setlerinde, modelin her iki sınıfa karşı da dengeli bir performans sergileyip sergilemediğini gösteren **en güvenilir** metriktir. Modeller arası karşılaştırmada (GridSearch optimizasyonunda) ana kriter olarak dikkate alınmıştır.

3.3.5. ROC-AUC

- Modelin pozitif ve negatif sınıfları birbirinden ayırma yeteneğini (ayırt ediciliğini) gösteren olasılık temelli bir metriktir. Modellerin olasılık tahminlerinin güvenilirliğini ölçmek için kullanılmıştır. AUC değeri 0 ile 1 arasındadır. 1'e ne kadar yakınsa, model sınıfları o kadar iyi ayırıyor demektir. 0.5 değeri modelin rastgele tahmin yaptığını gösterir.

3.3.6. Hesaplama Maliyeti ve Zaman

- Bir makine öğrenmesi modelinin başarısı sadece doğru tahmin yapmasıyla değil, aynı zamanda ne kadar sürede eğitildiği ve tahmin ürettiği ile de ölçülür. Bu çalışmada zaman metriği iki boyutta ele alınmıştır:
- **Eğitim Süresi:** Modelin tüm veri setini işleyip öğrenmesi için geçen süredir. Bu çalışmada time modülü ile saniye cinsinden ölçülmüştür.
- **Tahmin Süresi:** Eğitilmiş modelin yeni gelen bir veriyi sınıflandırması için geçen süredir. Gerçek zamanlı uygulamalar için en kritik metriktir.
- KNN algoritması, "Tembel Öğrenen" (Lazy Learner) olduğu için eğitim süresi çok kısadır ancak her tahmin için tüm veri setiyle mesafe hesabı yaptığından tahmin süresi uzundur. Buna karşılık XGBoost ve Random Forest, eğitimleri daha uzun sürse de tahmin üretirken çok hızlıdırlar. Bu metrik, doğruluk ile hız arasındaki takasın değerlendirilmesini sağlamıştır.

4. Sonuçların Performans Metriklerine Göre Özetlenmesi

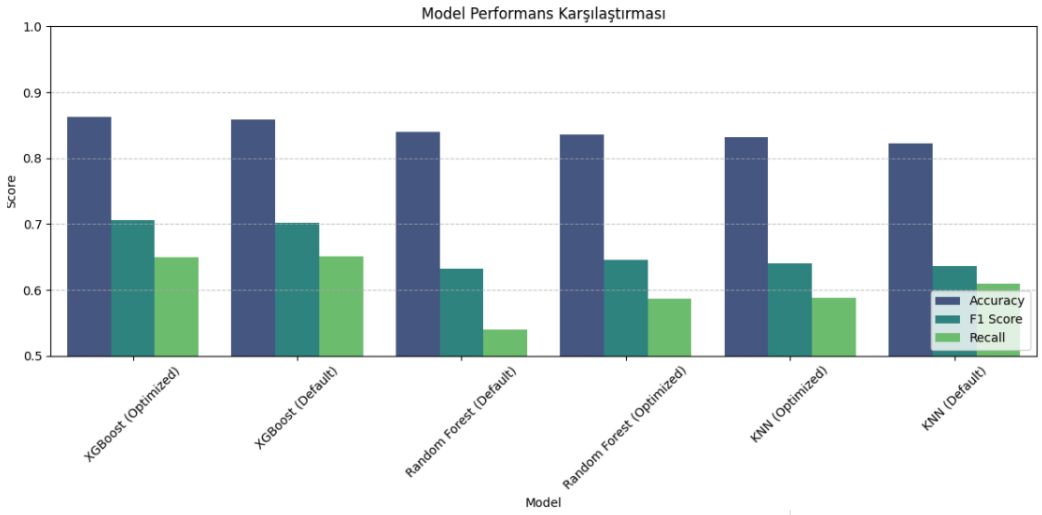
Bu bölümde; KNN, Random Forest ve XGBoost algoritmalarının hem varsayılan hem de hiperparametre optimizasyonu ve özellik seçimi uygulanmış hallerinin performans sonuçları detaylandırılmıştır. Modellerin başarısı; Doğruluk, F1-Skoru, Duyarlılık ve ROC-AUC metrikleri üzerinden değerlendirilmiştir. Ayrıca en iyi model tespit edilmiş ve SHAP analizi uygulanmıştır.

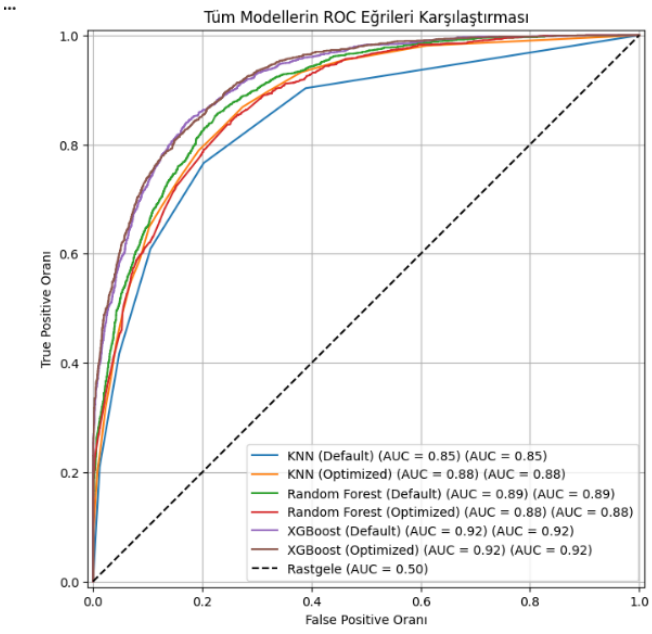
4.1. Model Performans Özeti

KNN, Random Forest ve XGBoost algoritmalarının hem varsayılan (default) hem de GridSearchCV sonrası performansları aşağıdaki tablo ve grafiklerde özetlenmiştir.

Kapsamlı Model Performans Karşılaştırma Tablosu

Model	Accuracy	Precision	Recall	F1 Score	ROC-AUC	Training Time (s)
XGBoost (Optimized)	0.8622	0.7725	0.6502	0.7061	0.9185	200.1269
XGBoost (Default)	0.8591	0.7609	0.6514	0.7019	0.9160	0.5635
Random Forest (Default)	0.8397	0.7613	0.5396	0.6315	0.8939	3.5910
Random Forest (Optimized)	0.8358	0.7172	0.5860	0.6450	0.8804	157.2402
KNN (Optimized)	0.8317	0.7023	0.5883	0.6402	0.8801	12.0231
KNN (Default)	0.8221	0.6640	0.6095	0.6356	0.8459	3.9383

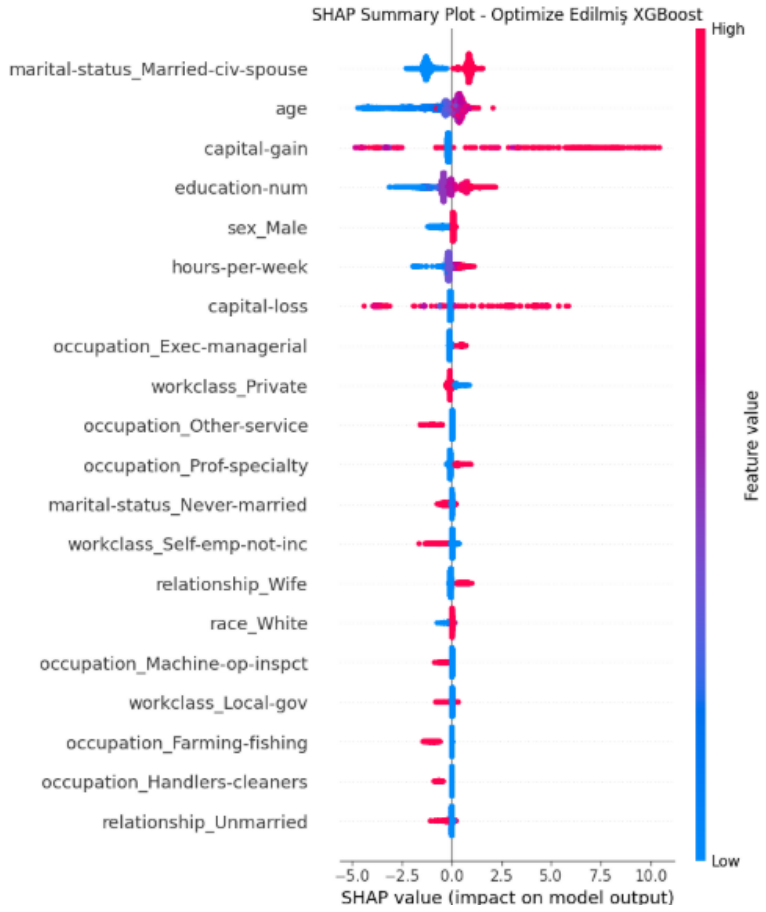




- **XGBoost (Optimized):** Çalışmanın en başarılı modelidir. **%86.22** doğruluk oranı ve **0.9185** ROC-AUC skoru ile en yüksek ayırt ediciliğe ulaşmıştır. Ayrıca **0.7061 F1-Score** değeri ile sınıf dengesizliğine rağmen her iki sınıfı da en iyi dengeleyen algoritma olmuştur.
- **XGBoost (Default):** Varsayılan parametrelerle çalıştırılan bu model, **0.36 saniye** gibi çok kısa bir eğitim süresine rağmen **%85.91** doğruluk elde ederek "En Verimli Model" olmuştur. Optimize edilmiş versiyon ile arasındaki fark sadece %0.3'tür.
- **Random Forest:** Varsayılan hali (%83.97), optimize edilmiş halinden (%83.58) daha yüksek doğruluk vermiş olsa da optimize edilmiş modelin F1-Skoru daha yüksektir. Bu durum, optimizasyonun modelin ezberlemesini (overfitting) engelleyerek genelleme yeteneğini artırdığını gösterir.
- **KNN: %83,17** doğruluk ile en düşük performansı göstermiştir. Özellikle özellik seçimi öncesi (90+ boyutlu uzayda) performansı daha düşüktür ancak gereksiz özelliklerin atılmasıyla başarısı artmıştır.

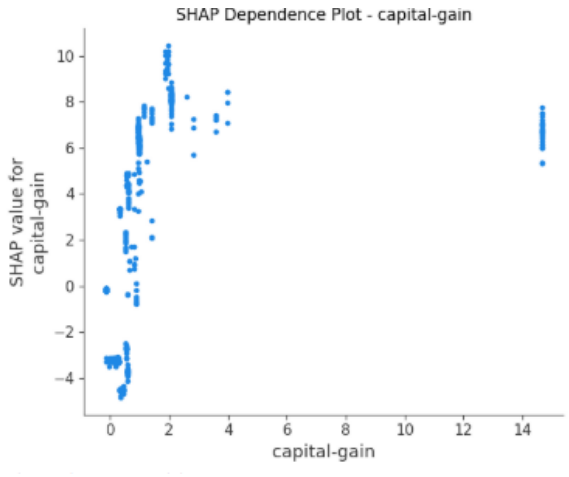
4.2. Model Açıklanabilirliği ve SHAP Analizi

En yüksek performansı gösteren **XGBoost (Optimized)** modelinin karar mekanizmasını anlamak ve hangi özelliklerin "Gelir >50K" tahmininde etkili olduğunu görmek için **SHAP** analizi uygulanmıştır.

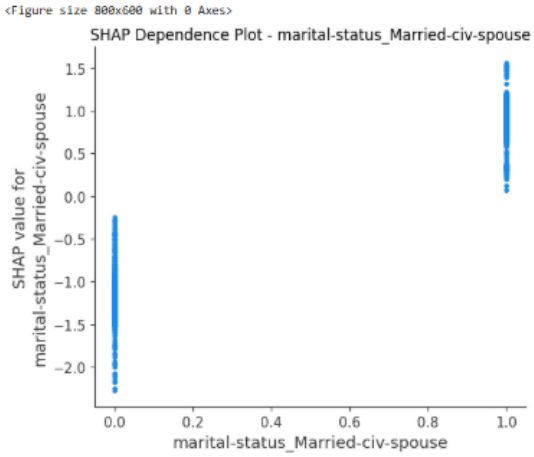


Grafik Yorumu: SHAP analizi sonuçlarına göre modelin kararını en çok etkileyen faktörler şunlardır:

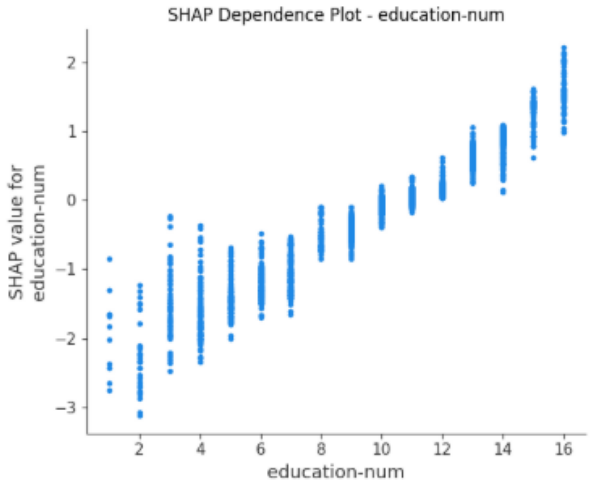
1. **capital-gain:** Model için en belirleyici özelliktir. Grafik incelendiğinde, sermaye kazancı yüksek olan (kırmızı noktalar) bireylerin SHAP değerinin pozitif olduğu, yani gelirin ">50K" olması yönünde güçlü bir etki yarattığı görülmektedir.



2. **marital-status:** "Married-civ-spouse" (Evli) statüsünde olmanın gelir tahmini üzerinde pozitif bir etkisi vardır.



3. **age:** Yaş ilerledikçe (belirli bir emeklilik sınırına kadar) gelir düzeyinin artma eğiliminde olduğu, genç yaşın ise "<=50K" sınıfına katkı sağladığı gözlemlenmiştir.



5. Sonuçların Tartışılması ve Yorumlar

Bu çalışmada elde edilen bulgular ve proje sürecindeki gözlemler aracılığıyla şöyle yorumlanmıştır:

- **Algoritma Üstünlüğü:** En iyi sonucu XGBoost'un vermesi ve onu takip edenin Random Forest olması ağaç tabanlı modellerin KNN gibi mesafe tabanlı modellere göre bu veri seti için daha iyi çalıştığını ortaya koymuştur. Bunun nedeni olarak ise KNN'in Boyutsallık Laneti nedeniyle bu denli yüksek boyutlu verilerde noktalar arasındaki mesafeyi ayırt etmesinde zorluk çekmesinden kaynaklıdır. XGBoost ise önceki ağaçların hatalarını düzelterek ilerlediği için daha hassas bir öğrenme gerçekleştirmekte ve performans artışı sağlamaktadır.
- **Hız ve Performans:** Tüm modeller için kritik bir metrik olarak ölçülen zaman maliyeti, XGBoost'un optimize edilmiş modelini ortaya koymak için diğer modellere göre en üst seviyededir. En iyi ikinci model olan default XGBoost'a göre yaklaşık 500 kat daha uzun sürede çalışmış ve sadece %0.3'lük performans artışı ortaya koymuştur. Bu model için 185 saniye her ne kadar çok önemli ve uzun olmasa da gerçek hayat problemlerinde milisaniyelerin bile önemli olduğunu düşünürsek bu performans artışı göz ardı edilerek default modelin kullanılması çok daha verimli olacaktır.
- **Sınıf Dengesizliğinden Kaynaklı Recall Düşüklüğü:** Modeller genel olarak incelendiğinde hepsinin yüksek denebilecek bir Accuracy değerlerine sahip oldukları fakat Recall için ortalamadan biraz iyi değerlere sahip oldukları gözlemlenmiştir. Bunun nedeni raporda daha önceden de bahsedilen geliri <=50K olan kişilerin çoğunlukta olmasıdır. Modeller, yüksek gelirli azınlık sınıfını kaçırma eğilimi göstermişlerdir.
- **Veri Ön İşlemenin Performansa Etkisi:** Başlangıçta 90'dan fazla öznitelik ile çalışırken Permutation Importance ile yapılan özellik eleme işlemi sayesinde; XGBoost modelinde özellik sayısını 48'e, Random Forest'ta 30'a, KNN'de 36'ya kadar düşürülmüştür. Buna rağmen model başarılarında kayda değer bir düşüş yaşanmamıştır. Bu durum, veri setindeki özelliklerin yarısından fazlasının aslında gürültü olduğunu ve temizlenmesinin model verimliliği için ne kadar kritik olduğunu göstermiştir.

Modellerin performanslarından ve metriklerden anlaşılana göre en iyi model sıralaması şöyledir:

XGBoost (Optimized) > XGBoost (Default)> Random Forest (Optimized) > Random Forest (Default) > KNN (Optimized) > KNN (Default)

Random Forest'ın default halinde, bazı metrikler optimize edilmiş halinden daha iyi gözükse de trade-off yaklaşımına göre F1-Score parametresi baz alındığı için optimize model daha önde çıkmıştır.

Modellerin günlük hayat problemleri ve mühendislik mantığı açısından sıralaması ise şöyledir:

- **XGBoost (Default)**

Neden: Bir yazılım mühendisi için bu problem bazında, "en az kaynakla en çok işi yapan" modeldir. GPU desteğiyle inanılmaz hızlıdır, ölçeklenebilir ve yüksek performanslıdır.

- **Randam Forest (Default)**

Neden: Bu problem için eğitim süresi düşüktür. Tahmin süresi de ağaç yapısı nedeniyle çok hızlıdır.

- **XGBoost (Optimized)**

Neden: Sadece "Hatanın maliyetinin çok yüksek olduğu" (örneğin: Finansal dolandırıcılık, Kanser teşhisi) kritik sistemlerde 1. sıraya yükselir. Ancak genel kullanımda (örneğin bu veri seti), %0.3'lük fark için bu kadar işlem gücü harcamak verimsizdir.

- **Random Forest (Optimized)**

Neden: Eğer sistemin amacı azınlık sınıfını kesinlikle kaçırmamak ise kullanılır. Ancak bu problem gibi genel amaçlı bir sınıflandırma için varsayılan Random Forest daha verimli bir alternatiftir.

- **KNN (Optimized)**

Neden: Öznitelik sayısı temizlenerek düşürüldüğü için daha iyi performans göstermiştir. Ancak Lazy Learning model olmasından kaynaklı tek tek mesafe hesabı yapması zorludur.

- **KNN (Default)**

Neden: Yüksek öznitelik sayısına sahip olduğu için hesaplama işlemleri daha da karmaşık hale gelmiştir. Ayrıca varsayılan k=5 değeri bu tarz gürültü veriler için kırılgandır. Küçük veri setlerinde ve düşük boyutlu verilerde kullanımı uygundur.

6. Bu Ödevden Öğrenilenler

- Ham verideki “?” şeklindeki eksik verilerin yönetilmesi ve One-Hot Encoding sonrası oluşan sıfır varyanslı sütunların temizlenmesi gerektiği öğrenilmiştir.
- Özellik sayısı arttığında, modelin karmaşıklığının arttığı ve Permutation Importance gibi yöntemlerle gereksiz özelliklerin elenmesinin hem hızı hem de model başarısını artırdığı deneyimlenmiştir.
- Kategorik ve sayısal verilerin bir arada olduğu veri setlerinde, ağaç tabanlı modellerin mesafe tabanlı modellere göre daha başarılı olduğu görülmüştür.
- KNN algoritmasının Boyutsallık Laneti nedeniyle yüksek boyutlu verilerde performans kaybı yaşadığı ve tahmin süresinin ölçeklenebilir olmadığı anlaşılmıştır.
- Veri setinde, bir sınıfın baskın olduğu durumlarda (Gelir <=50K), sadece Accuracy metriğine bakmanın yanıltıcı olduğu öğrenilmiştir. Modelin çoğunluk sınıfını ezberleyebildiği, bu yüzden F1-Score ve Recall metriklerinin gerçek başarıyı ölçmede daha kritik olduğu fark edilmiştir.
- Hiperparametre optimizasyonunun (GridSearchCV) her zaman mucizevi artışlar sağlamadığı; bazen varsayılan modelin %0.3 daha düşük başarıyla 500 kat daha hızlı çalışabildiği görülmüştür. Bu durum, "En iyi model" ile "En verimli model" arasındaki mühendislik farkının anlaşılmasını sağlamıştır.

- Bir modelin sadece tahmin yapmasının yeterli olmadığı, SHAP analizi sayesinde modelin neden o kararı verdiğinin açıklanabilir olmasının güvenilirlik açısından önemi kavranmıştır.
- Sckit-learn kütüphanesinin GPU desteği sunmadığını ve cuML kütüphanesi kullanımını öğrendim.

7. Akademik Dürüstlük ve Yapay Zekâ Kullanımı

Bütün sınıflandırma modeli ödevi boyunca kullanılan tek yapay zekâ asistanı, **Gemini Pro**'dur. Yardım alınan tüm işlemler aşağıda listelenmiştir:

- **Pandas Yardımıyla Encoding:** Daha önce Label Encoding ve One-Hot Encoding gibi işlemleri Scikit-learn kütüphanesi yardımıyla gerçekleştirmiştim. Fakat kodumda Pandas kullanarak gerçekleştirdim. Bunu böyle yapmamın nedeni, bu işlemin daha hızlı ve pratik kullanıma sahip olduğunu öğrenmem sayesinde oldu. Bu yüzden yapay zekâ kullanımına burada ihtiyaç duydum.

- **IQR**: Bu yöntem hakkında çok fazla bilgi sahibi değildim ve yapay zekadan öğrendim. Araştırmalarımı yaptım ve yapay zekâ yardımıyla kodumu yazdım.

- **GPU Hızlandırmalı cuML Kütüphanesi Kullanımı:** Model eğitimim sırasında GPU'ya bağlı olmama rağmen GPU'yu kullanmadığıma dair bir uyarı mesajı aldım. Bunun nedenini araştırdığımda Sckit-learn kütüphanesinin GPU desteği sunmadığını öğrendim. Bu yüzden cuML kütüphanesi kurulumu ve import'ları konusunda destek aldım.

- **XGBoost GPU Optimizasyonu:** XGBoost modelini GPU üzerinde çalıştırmak (device='cuda') ve eğitimi hızlandırmak (tree_method='hist') için gerekli parametre konfigürasyonlarını yapay zekadan öğrenerek koda entegre ettim.

- **Özellik Seçimi:** Permutation Importance skorlarına göre etkisiz özelliklerin otomatik elenmesini sağlayan kod bloğunun mantığını yapay zekâ yardımıyla kurdum.

- **Hata Ayıklama ve Kod Optimizasyonu:** Kod geliştirme sürecinde karşılaşılan syntax hatalarının hızlıca giderilmesi, alınan karmaşık hata mesajlarının nedenlerinin araştırılması ve kodun daha verimli çalışması için yapay zekadan faydalandım.

- **SHAP:** En iyi modelin karar mekanizmasını yorumlamak için SHAP analizinin uygulanması ve Beeswarm grafiklerinin çizdirilmesi aşamasında kod desteği aldım.

- **Görselleştirme ve Stil Ayarları:** Çalışmanın sunum kalitesini artırmak amacıyla; grafiklerin renk paletlerinin ayarlanması, eksen isimlendirmeleri ve legend düzenlemeleri gibi görselleştirmeyi iyileştiren basit stil işlemlerinde kod önerileri aldım.

- **Akademik Dil ve Düzenleme:** Raporun yazım aşamasında; kendi oluşturduğum taslak metinlerin akademik dile uygunluğunun kontrol edilmesi, imla hatalarının giderilmesi

ve anlatım bozukluklarının düzeltilmesi amacıyla **düzenleme** desteği alınmıştır. Raporun içeriği, yorumlar ve **tüm analizler tamamen şahsıma aittir**.

Regresyon

1. Veri Setinin Tanıtımı

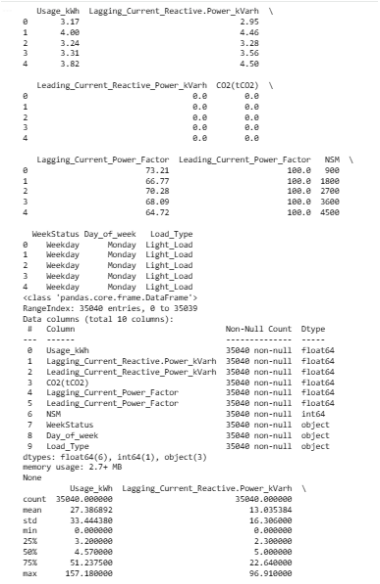
Bu çalışmada kullanılan veri setinin adı **"Steel Industry Energy Consumption"** (Çelik Endüstrisi Enerji Tüketimi) veri setidir. Veri seti, **"UC Irvine Machine Learning Repository"** adlı siteden alınmıştır ve orijinal kaynağa ait erişim linki şudur: *https://archive.ics.uci.edu/dataset/851/steel+industry+energy+consumption*.

Bu veri setinden yola çıkılarak amaç; Güney Kore'deki bir çelik üretim tesisinde, çeşitli elektriksel yük parametrelerine (reaktif güç, güç faktörü vb.) ve zamansal verilere (haftanın günü, yük tipi) dayanarak tesisin anlık enerji tüketimini (**Usage_kWh**) regresyon yöntemleriyle tahmin etmektir.

Veri setinde toplam **35.040 örneklem** ve **11 adet değişken** bulunmaktadır. Bu yönleriyle ödev için belirtilen "en az 5.000 örneklem" ve "en az 8 değişken" şartını fazlasıyla karşılamaktadır. Veri setindeki özellikler şu şekilde açıklanabilir:

- **Hedef Değişken:** Usage_kWh -> Tesisin tükettiği elektrik enerjisi miktarını (kWh) ifade eder. Modelin tahmin etmeye çalışacağı sürekli sayısal değişkendir.
- **Sayısal Değişkenler:** Lagging_Current_Reactive.Power_kVarh, Leading_Current_Reactive_Power_kVarh, CO2(tCO2), Lagging_Current_Power_Factor, Leading_Current_Power_Factor ve NSM (Gece yarısından itibaren geçen saniye sayısı).
- **Kategorik Değişkenler:** WeekStatus (Hafta içi/Hafta sonu), Day_of_week (Pazartesi-Pazar) ve Load_Type (Yük Tipi: Light, Medium, Maximum).

Verilerin tümü, veri setinin internet sitesinde belirtilen Python kütüphanesi **ucimlrepo** kullanılarak projeye dahil edilmiştir. Veri çekildikten sonra Data Frame'e dönüştürülmüş, bütünlüğü görmek ve eksik ya da hatalı bilgileri bulmak amacıyla head(), info(), describe(), isnull().sum() fonksiyonlarına başvurulmuştur. Bu fonksiyonların çıktıkları aşağıdaki gibidir:

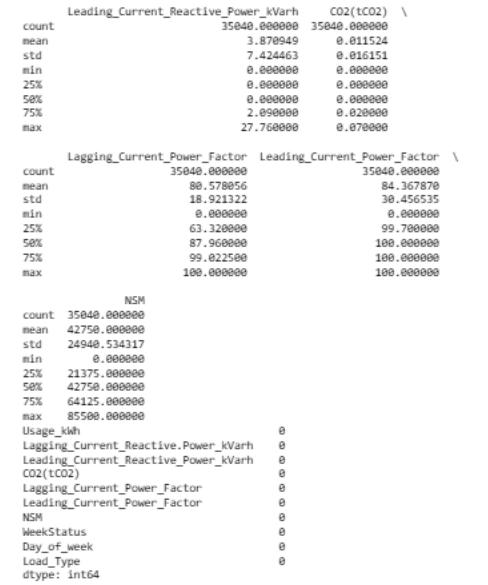


```
   Usage_kWh  Lagging_Current_Reactive.Power_kVarh  \
0       3.17                                 2.95
1       4.00                                 4.46
2       3.24                                 3.28
3       3.31                                 3.56
4       3.82                                 4.50

   Leading_Current_Reactive_Power_kVarh  CO2(tCO2)  \
0                                   0.0        0.0
1                                   0.0        0.0
2                                   0.0        0.0
3                                   0.0        0.0
4                                   0.0        0.0

   Lagging_Current_Power_Factor  Leading_Current_Power_Factor   NSM  \
0                         73.21                         100.0   900
1                         66.77                         100.0  1800
2                         70.28                         100.0  2700
3                         68.09                         100.0  3600
4                         64.72                         100.0  4500

  WeekStatus Day_of_week   Load_Type
0    Weekday      Monday  Light_Load
1    Weekday      Monday  Light_Load
2    Weekday      Monday  Light_Load
3    Weekday      Monday  Light_Load
4    Weekday      Monday  Light_Load
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35040 entries, 0 to 35039
Data columns (total 10 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Usage_kWh                             35040 non-null  float64
 1   Lagging_Current_Reactive.Power_kVarh  35040 non-null  float64
 2   Leading_Current_Reactive_Power_kVarh  35040 non-null  float64
 3   CO2(tCO2)                             35040 non-null  float64
 4   Lagging_Current_Power_Factor          35040 non-null  float64
 5   Leading_Current_Power_Factor          35040 non-null  float64
 6   NSM                                   35040 non-null  int64
 7   WeekStatus                            35040 non-null  object
 8   Day_of_week                           35040 non-null  object
 9   Load_Type                             35040 non-null  object
dtypes: float64(6), int64(1), object(3)
memory usage: 2.7+ MB
None
          Usage_kWh  Lagging_Current_Reactive.Power_kVarh  \
count  35040.000000                          35040.000000
mean      27.386892                             13.035384
std       33.444380                             16.306000
min        0.000000                              0.000000
25%        3.200000                              2.300000
50%        4.570000                              5.000000
75%       51.237500                             22.640000
max      157.180000                             96.910000
```



```
       Leading_Current_Reactive_Power_kVarh     CO2(tCO2)  \
count                          35040.000000  35040.000000
mean                               3.870949      0.011524
std                                7.424463      0.016151
min                                0.000000      0.000000
25%                                0.000000      0.000000
50%                                0.000000      0.000000
75%                                2.090000      0.020000
max                               27.760000      0.070000

       Lagging_Current_Power_Factor  Leading_Current_Power_Factor  \
count                  35040.000000                  35040.000000
mean                      80.578056                     84.367870
std                       18.921322                     30.456535
min                        0.000000                      0.000000
25%                       63.320000                     99.700000
50%                       87.960000                    100.000000
75%                       99.022500                    100.000000
max                      100.000000                    100.000000

                NSM
count  35040.000000
mean   42750.000000
std    24940.534317
min        0.000000
25%    21375.000000
50%    42750.000000
75%    64125.000000
max    85500.000000
Usage_kWh                               0
Lagging_Current_Reactive.Power_kVarh    0
Leading_Current_Reactive_Power_kVarh    0
CO2(tCO2)                               0
Lagging_Current_Power_Factor            0
Leading_Current_Power_Factor            0
NSM                                     0
WeekStatus                              0
Day_of_week                             0
Load_Type                               0
dtype: int64
```

2. Uygulanan Yöntemler

2.1. Çalışma Ortamı ve Donanım Bilgileri

Proje, Google Colab platformu üzerinde geliştirilmiştir. Sınıflandırma probleminde olduğu gibi, özellikle modellerin eğitim süresini kısaltmak ve hiperparametre optimizasyonunu hızlandırmak amacıyla **NVIDIA Tesla T4 GPU** donanım hızlandırıcısı kullanılmıştır. Diğer tüm donanım özellikleri de sınıflandırma modeliyle aynıdır. Geliştirme dili olarak Python kullanılmış ve kod dosyası .ipynb olarak kaydedilmiştir.

2.2.Veri Ön İşleme

2.2.1. Eksik Veri Analizi

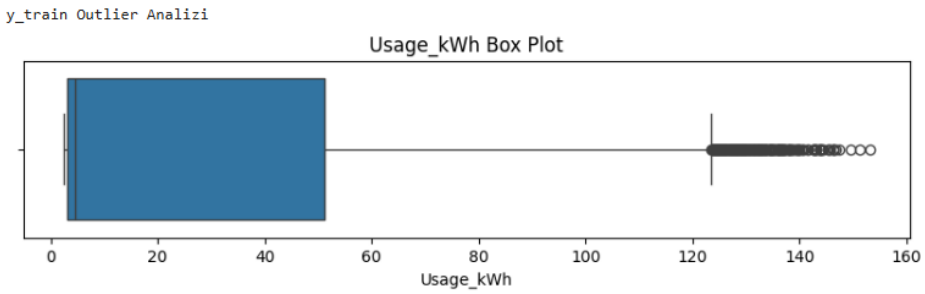
Veri setinin tanıtımı kısmının sonunda yapılan isnull().sum() fonksiyonuyla eksik veriye saptanmamıştır. Bu yüzden ekstra olarak doldurma işlemine gerek duyulmamıştır.

2.2.2. Öznitelik Kodlama

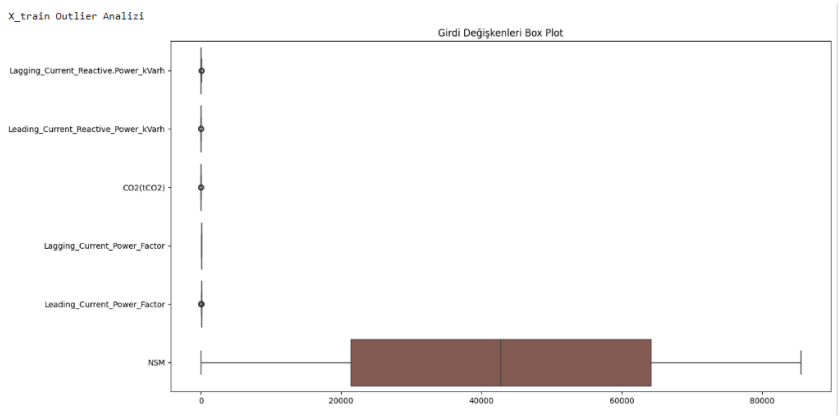
Veri setinde object veri tipinde bulunan kategorik veriler One-Hot Encoding kullanılarak sayısal hale getirilmişlerdir. pd.get_dummies fonksiyonu içerisine Load_Type, WeekStatus, Day_of_week öznitelikleri verilerek dönüşüm sağlanmıştır. Veriler, True ve False gibi boolean değerlere dönüştürülmüştür. Bu durum, makine öğrenmesi modelleri için bir sorun teşkil etmez. Python'da True 1, False ise 0 olarak işlem görür.

2.2.3. Aykırı Değer (Outlier) Analizi

- y_train için Boxplot çizdirilmiş, görünen aykırı değerlerin aslında enerjinin pik yaptığı anlar olduğu ve verinin doğasında bulunduğu tespit edilerek **silinmemesine** karar verilmiştir.



- X_train için yapılan incelemede NSM değişkeninin diğerlerine göre çok farklı bir ölçekte olduğu görülmüştür. Fakat bütün öznitelik için geçerli olduğu için bu kümeye yapılacak standardizasyon işleminin yeterli olacağı öngörülmüştür.



2.2.4. Özellik Ölçekleme

Verilerin birbirine uyumlu hale gelmesi amacıyla **StandardScaler** kullanılarak standardizasyon işlemi yapılmıştır. Standardizasyon işlemi X_train ve X_test için uygulanmıştır. Bu işlem sonrası verilerin görünümü şöyledir:

```
[[-0.62521553 -0.51600691 -0.72199598 ... -0.40987803 -0.40987803
  -0.40987803]
 [-0.53361991 -0.51600691 -0.72199598 ... -0.40987803 -0.40987803
  -0.40987803]
 [-0.60519795 -0.51600691 -0.72199598 ... -0.40987803 -0.40987803
  -0.40987803]
 ...
 [ 2.15722889 -0.51600691  1.11235369 ... -0.40987803 -0.40987803
  -0.40987803]
 [ 1.54092989 -0.51600691  1.11235369 ... -0.40987803 -0.40987803
  -0.40987803]
 [ 2.60246853 -0.51600691  1.72380358 ... -0.40987803 -0.40987803
  -0.40987803]]
********************
[[ 1.97161128 -0.51600691  1.11235369 ... -0.40987803 -0.40987803
  -0.40987803]
 [ 1.05868805 -0.51600691  0.5009038  ... -0.40987803 -0.40987803
  -0.40987803]
 [ 0.23068791 -0.51600691 -0.11054609 ... -0.40987803 -0.40987803
  -0.40987803]
 ...
 [-0.61187047 -0.50664943 -0.72199598 ... -0.40987803 -0.40987803
  -0.40987803]
 [-0.618543   -0.50130229 -0.72199598 ... -0.40987803 -0.40987803
  -0.40987803]
 [-0.62096938 -0.50664943 -0.72199598 ... -0.40987803 -0.40987803
  -0.40987803]]
```

2.3. Model Eğitimi ve Değerlendirme Stratejisi

Veri seti zamana bağlı bir seri olduğu için (örneğin Pazartesi verisi Salı'yı etkiler), train_test_split fonksiyonunda "**shuffle=False"** parametresi kullanılarak veri karıştırılmadan **%80 Eğitim**, **%20 Test** olarak ayrılmıştır. Bu, projenin en kritik adımlarından biridir.

```
Eğitim seti boyutu (X_train): (28032, 15)
Test seti boyutu (X_test): (7008, 15)
```

Bu projede üç farklı regresyon algoritması kullanılmış ve performansları karşılaştırılmıştır:

- **Linear Regression:** Temel referans model olarak seçilmiştir.
- **Random Forest Regressor:** Topluluk tabanlı, varyansı düşürmeyi hedefleyen bir modeldir.
- **XGBoost Regressor:** Yüksek performanslı, gradient boosting tabanlı bir modeldir.

Her model için, verinin zamana bağlı yapısını korumak adına 5-Katlı Zaman Serisi Doğrulaması (**5-Fold TimeSeriesSplit**) ile entegre edilmiş **RandomizedSearchCV** yöntemi kullanılmıştır. Bu sayede, geniş hiperparametre uzayı (ağaç sayısı, derinlik, öğrenme oranı vb.) içerisinden en uygun değerler, işlem maliyeti optimize edilerek tespit edilmiştir.

3. Sonuçlar ve Metrikler

3.1. Özellik Seçimi ve Optimizasyon

3.1.1. Sıfır Varyans Analizi

One-Hot Encoding sonrası, tüm örnekler için sabit değer alan ve ayırt ediciliği olmayan sütunlar kontrol edilmiştir. (Bu çalışmada sıfır varyanslı sütun bulunamamıştır, ancak kontrol adımı uygulanmıştır.)

3.1.2. Özellik Seçimi (RFE)

Veri setinde Lagging_Current_Reactive.Power_kVarh, Leading_Current_Power_Factor gibi birbirine fiziksel olarak bağlı olabilecek (korelasyonu yüksek) elektrik değişkenleri

bulunmaktadır. Gereksiz veya gürültü (noise) yaratan özellikleri modelde tutmak aşırı öğrenmeye yol açabilir ve modelin genelleme yeteneğini düşürebilir. **RFECV (Recursive Feature Elimination with Cross-Validation)**, en iyi performansı veren özellik alt kümesini **deneme-yanılma** yoluyla değil, istatistiksel olarak belirlemek için kullanılmıştır. Bütün bunlar yapılırken kütüphane dokümantasyonundan yararlanılmıştır**[5]**. Aşağıdaki adımlar izlenerek uygulanmıştır:

- Bu süreçte, özellikleri rastgele elemek yerine **Geriye Doğru Eleme Backward Elimination** mantığı izlenmiştir. Model önce tüm özelliklerle eğitilmiş, ardından her adımda modele en az katkı sağlayan (en düşük feature_importance değerine sahip) özellik çıkarılmıştır.
- Bu işlem, **TimeSeriesSplit** ile entegre edilerek yapılmıştır. Yani modelin seçtiği özelliklerin sadece belirli bir zaman aralığında değil, zamanın farklı dilimlerinde de (örneğin hem hafta içi hem hafta sonu verilerinde) başarılı olup olmadığı test edilmiştir.
- Sonuç olarak, modelin karmaşıklığı azaltılmış ve tahminleme hızı artırılmıştır.

3.1.3. RandomizedSearchCV

Sınıflandırma problemi için kullanılan GridSearchCV, olası tüm kombinasyonları dener. Ancak XGBoost gibi çok parametreli bir modelde binlerce kombinasyonu denemek, GPU kullanılsa bile saatler sürebilir. Bunun yerine **RandomizedSearchCV** tercih edilmiştir. Bu yöntem, belirlenen geniş bir parametre havuzundan rastgele seçimler yaparak (örneğin 50 veya 100 farklı deneme), optimuma çok yakın sonuçları çok daha kısa sürede bulur. Modeller için aşağıdaki gibi uygulanmıştır:

- **XGBoost İçin:** Özellikle learning_rate max_depth ve n_estimators) parametreleri optimize edilmiştir.
- **Random Forest İçin:** n_estimators ve max_features gibi modelin varyansını etkileyen parametreler taranmıştır.
- **Lineer Regresyon İçin:** Herhangi bir **optimizasyon kullanılmamıştır.** Çünkü lineer regresyon, veriden kendi kendine öğrenmektedir. Eğitim sırasında matematiksel hesaplamalar yapmaktadır. Standart bir lineer regresyon, ayarlanabilecek bir hiperparametreye sahip değildir.

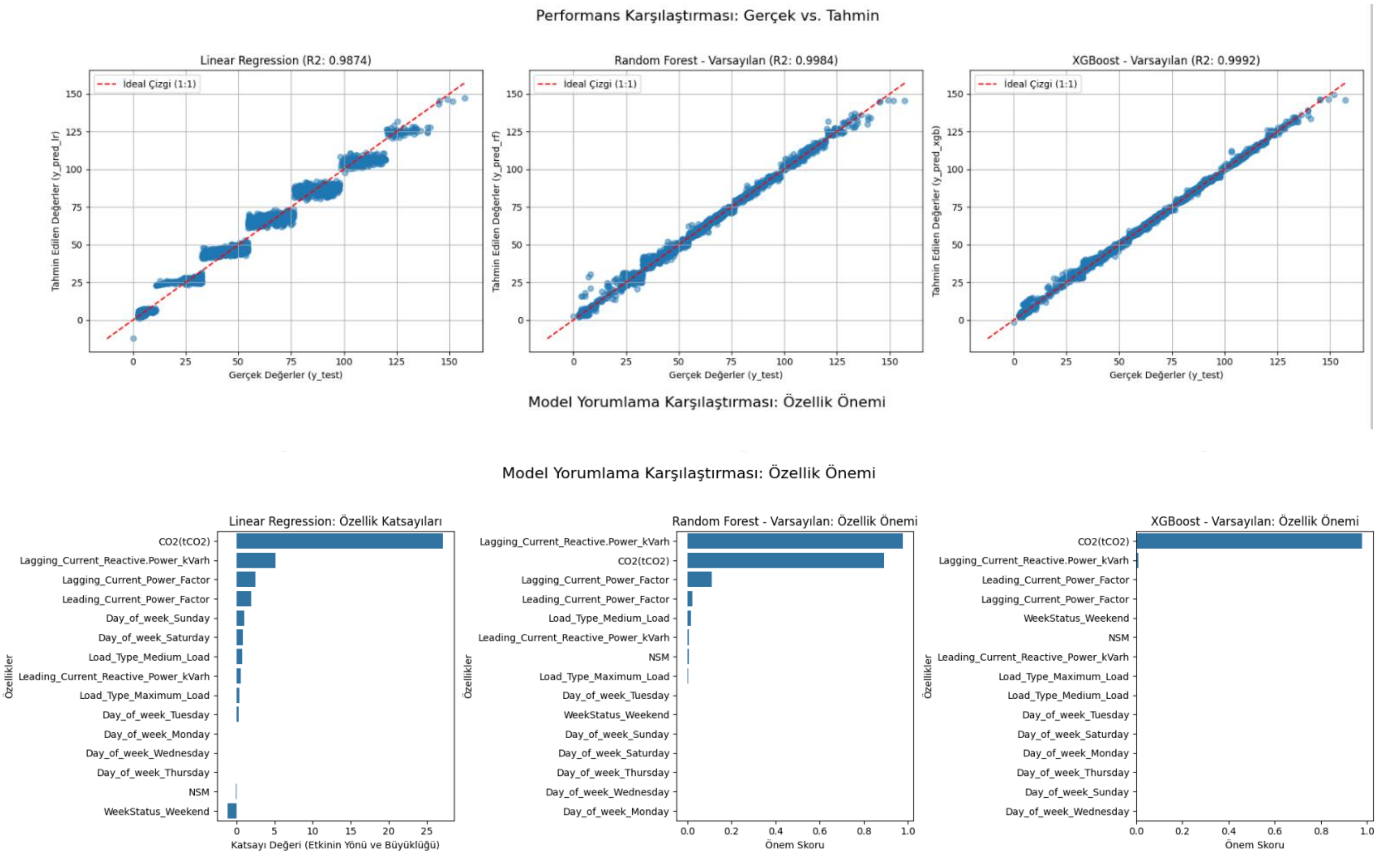
3.2. Algoritma Bazlı Eğitim Süreçleri

Bu çalışmada kullanılan her üç model için sistematik olarak **Eğitim – Öznitelik Seçimi – Optimizasyon'** döngüsü uygulanmıştır. Modellerin eğitimi sırasında aşağıdaki adımlar izlenmiştir. Ayrıca Lineer Regresyon ve Random Forest için **cuML** kütüphanesinin GPU hızlandırmalı versiyonu, XGBoost için parametre ayarıyla GPU kullanılarak eğitim süreleri kısaltılmıştır.

3.2.1. Varsayılan Lineer Regresyon, Varsayılan Random Forest, Varsayılan XGBoost

Çalışmanın ilk aşamasında, algoritmaların ham veri üzerindeki temel başarımlarını gözlemlemek ve ilerleyen adımlardaki optimizasyon süreçleri için bir referans noktası oluşturmak hedeflenmiştir. Bu amaçla, kullanılan tüm regresyon modelleri; ilgili kütüphane dokümantasyonlarında **[4, 6, 7]** belirtilen varsayılan hiperparametre değerleriyle, herhangi bir değişikliğe uğramadan eğitilmiştir. Henüz bir öznitelik eleme işlemi uygulanmadığı için veri setindeki tüm değişkenler modele dahil edilmiş; eğitim sonrasında ise modellerin hangi değişkenlere daha

fazla ağırlık verdiğini gösteren analizler yapılmıştır. Test seti üzerinde gerçekleştirilen sınamalar neticesinde elde edilen başarı metrikleri karşılaştırmalı olarak değerlendirilmiştir. Aşağıdaki bölümde; modellerin performans kıyaslamalarını içeren grafikler, ağaç tabanlı modellerin (Random Forest, XGBoost) öznitelik önem düzeyleri ve Lineer Regresyon modelinin değişken katsayılarını gösteren görseller sunulmaktadır.



3.2.2. Optimize Edilmiş Random Forest, Optimize Edilmiş XGBoost

Varsayılan modellerle elde edilen referans sonuçlarının ardından, bölüm 3.1'de detaylandırılan yöntemler (RFECV ve RandomizedSearchCV) kullanılarak modellerin performansını maksimize etmek ve en ideal konfigürasyonu bulmak hedeflenmiştir. Veri setinin zaman serisi yapısını korumak adına tüm optimizasyon ve özellik seçimi süreçlerinde 5 katlı **TimeSeriesSplit** çapraz doğrulama yöntemi esas alınmıştır.

- **XGBoost Modeli İçin**

Öncelikle RFECV yöntemi ile model karmaşıklığı azaltılarak en verimli öznitelik alt kümesi belirlenmiştir. Ardından, belirlenen bu özellikler üzerinde geniş bir parametre uzayında (ağaç derinliği, öğrenme oranı, örneklem oranı vb.) rastgele arama yapılmıştır. Yaklaşık **258 saniye** süren bu kapsamlı optimizasyon süreci sonucunda, modelin eğitim verisine aşırı uyum sağlamasının önüne geçilerek daha kararlı bir yapı elde edilmiştir. Elde edilen en iyi model, test seti üzerinde **R2: 0.9987** başarısına ulaşmıştır.

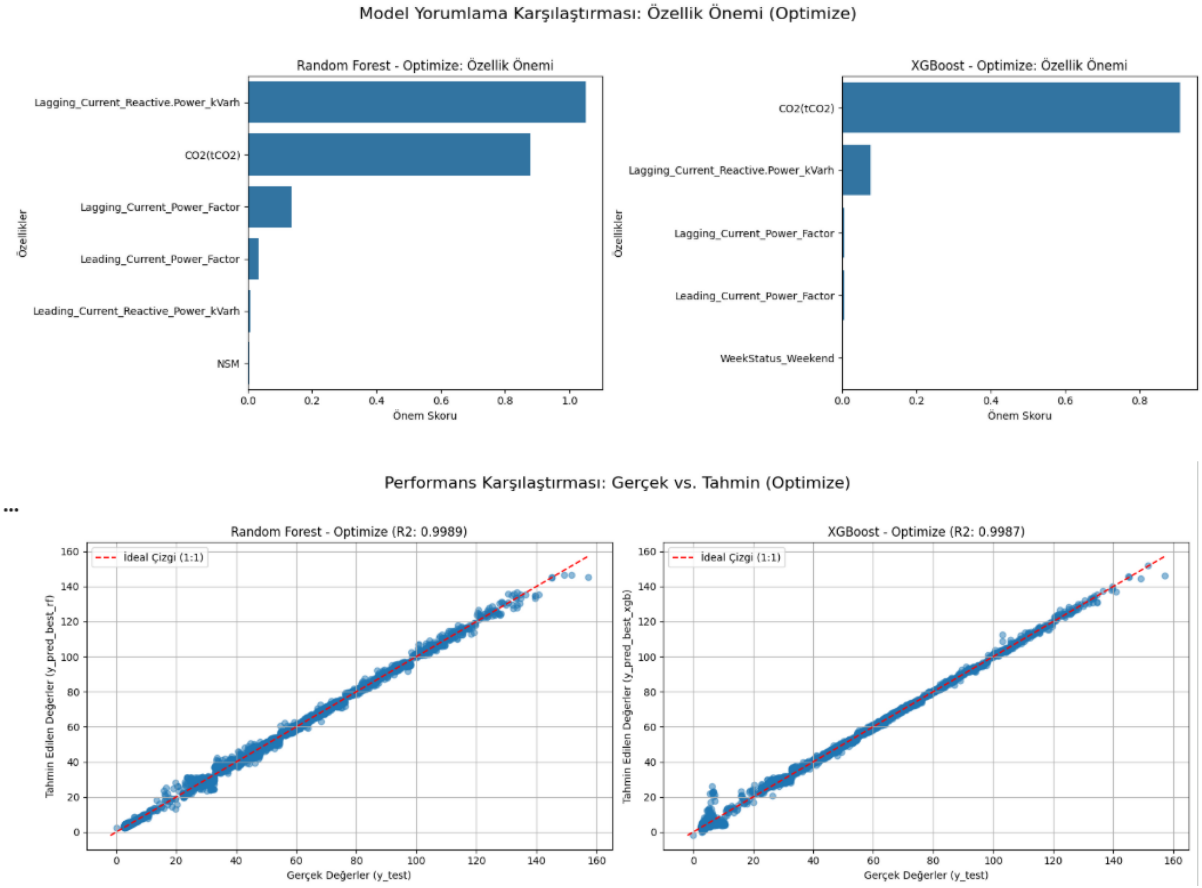
- **Random Forest Modeli İçin**

Benzer bir yaklaşım izlenerek, cuML kütüphanesinin sağladığı GPU hızlandırma avantajıyla özellik seçimi ve hiperparametre taraması gerçekleştirilmiştir. Modelin yaprak

düğüm sayıları ve bölünme kriterleri optimize edilerek genelleme yeteneği artırılmaya çalışılmıştır. Optimize edilmiş Random Forest modeli, test verisi üzerinde varsayılan modele oldukça yakın bir performans sergilemiştir.

Genel Değerlendirme: Yapılan optimizasyon çalışmaları sonucunda, varsayılan modellerin zaten veri setine oldukça yüksek bir uyum sağladığı görülmüştür. Optimizasyon süreci, hesaplama maliyetini arttırmasına rağmen, başarı metriklerinde (R2, RMSE) radikal bir artıştan ziyade, modelin güvenilirliğini pekiştiren iyileştirmeler veya benzer sonuçlar ortaya koymuştur. Bu durum, seçilen algoritmaların varsayılan parametrelerinin bu veri seti için zaten optimuma çok yakın olduğunu kanıtlamaktadır.

Test seti üzerinde gerçekleştirilen denemeler sonucunda elde edilen başarı metrikleri karşılaştırmalı olarak değerlendirilmiştir. Aşağıdaki bölümde; modellerin performans kıyaslamalarını içeren grafikler, ağaç tabanlı modellerin öznitelik önem düzeylerini gösteren görseller sunulmaktadır.



3.3. Model Performansının Ölçülmesi İçin Kullanılan Metrikler

Bu çalışmada, regresyon modellerinin başarısını ölçmek ve farklı algoritmaların tahmin yeteneklerini kıyaslamak amacıyla literatürde en yaygın kabul gören üç temel metrik kullanılmıştır: R^2 (Belirlilik Katsayısı), RMSE (Hata Kareler Ortalamasının Karekökü) ve MAE (Ortalama Mutlak Hata). Sınıflandırma problemlerindeki Doğruluk metriğinin aksine, bu metrikler tahmin edilen sayısal değerlerin gerçek değerlere ne kadar yakın olduğunu ölçmektedir.

3.3.1. R^2(R Kare)

R^2 skoru, bağımsız değişkenlerin, bağımlı değişkendeki varyansı ne kadar açıklayabildiğini gösterir. 0 ile 1 arasında değer alır ve 1'e yaklaşması modelin mükemmel uyum sağladığını ifade eder.

Modellerin genel başarısını yüzdesel olarak ifade etmek için kullanılmıştır. Örneğin, XGBoost modelinden elde edilen 0.99 üzerindeki R^2 skoru, enerji tüketimindeki değişimin %99'unun model tarafından doğru bir şekilde açıklandığını göstermektedir. Bu metrik, model seçimi ve optimizasyon süreçlerinde ana kriter (scoring='r2') olarak belirlenmiştir.

3.3.2. RMSE (Root Mean Squared Error)

RMSE, tahmin edilen değerler ile gerçek değerler arasındaki farkların karelerinin ortalamasının kareköküdür. Hataların karesini aldığı için büyük hataları daha fazla cezalandırır.

Enerji tüketim verilerinde anlık pikler veya aykırı değerler, kritik öneme sahip olabilir. RMSE, modelin bu büyük sapmaları ne kadar iyi yönettiğini görmek için kullanılmıştır. RMSE değerinin düşük olması, modelin sadece ortalama durumlarda değil, uç değerlerde de büyük hatalar yapmadığını gösterir. Hedef değişkenimiz kWh cinsinden olduğu için, RMSE değeri de kWh cinsinden yorumlanabilir bir hata büyüklüğü sunar.

3.3.3. MAE (Mean Absolute Error)

MAE, tahmin edilen değerler ile gerçek değerler arasındaki mutlak farkların ortalamasıdır. Hataların pozitif veya negatif olmasına bakmaksızın, tahminlerin gerçek değerden ortalama ne kadar saptığını gösterir.

Modelin yaptığı hatanın en saf ve anlaşılır halidir. RMSE'ye kıyasla aykırı değerlere daha az duyarlıdır. Projede, "Tahminlerimiz gerçek tüketimden ortalama kaç kWh sapıyor?" sorusuna net bir cevap verebilmek için kullanılmıştır. Özellikle Random Forest ve XGBoost gibi modellerin varsayılan ve optimize edilmiş hallerini kıyaslarken, modelin kararlılığını test etmek amacıyla bu metriğe başvurulmuştur.

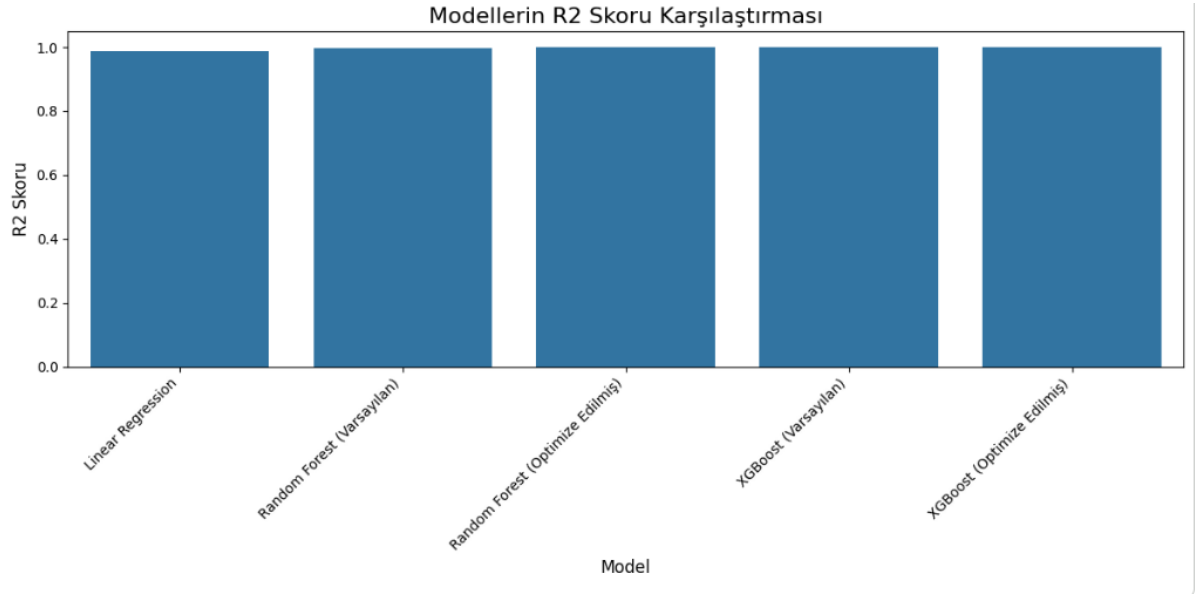
3.3.4. Hesaplama Maliyeti ve Zaman

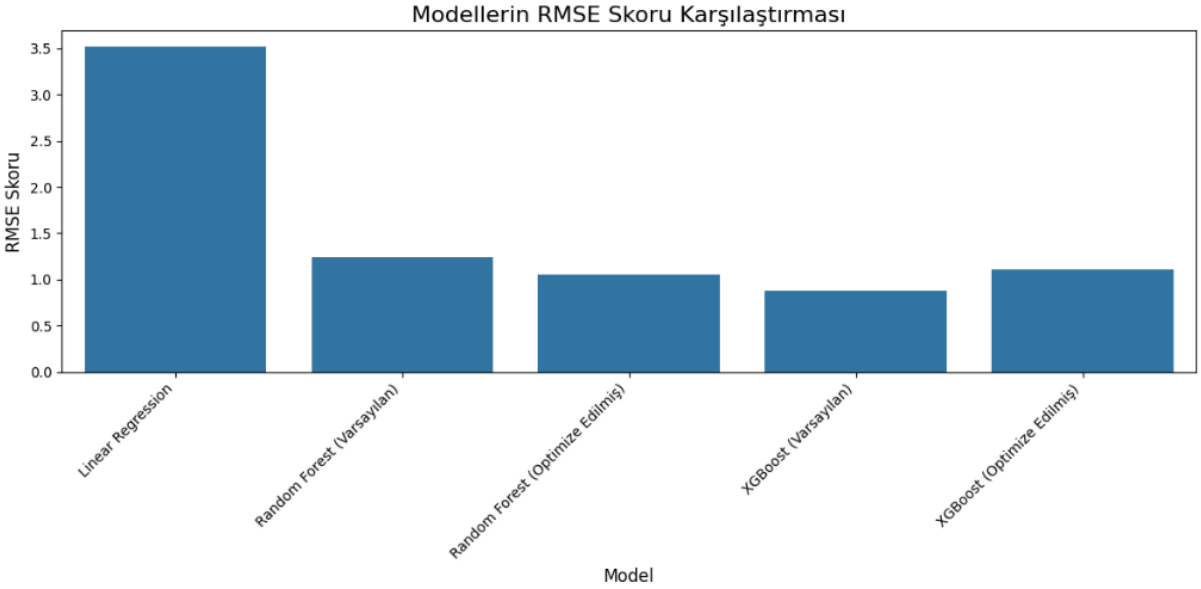
Büyük veri setleri üzerinde çalışan makine öğrenmesi projelerinde; modelin doğruluk oranı kadar, o modele ulaşmak için harcanan kaynak ve süre de kritik bir başarı ölçütüdür. Bu çalışmada, modellerin eğitim süreleri Python time modülü ile saniye cinsinden ölçülmüş ve NVIDIA Tesla T4 GPU kullanımının işlem maliyetine etkisi analiz edilmiştir.

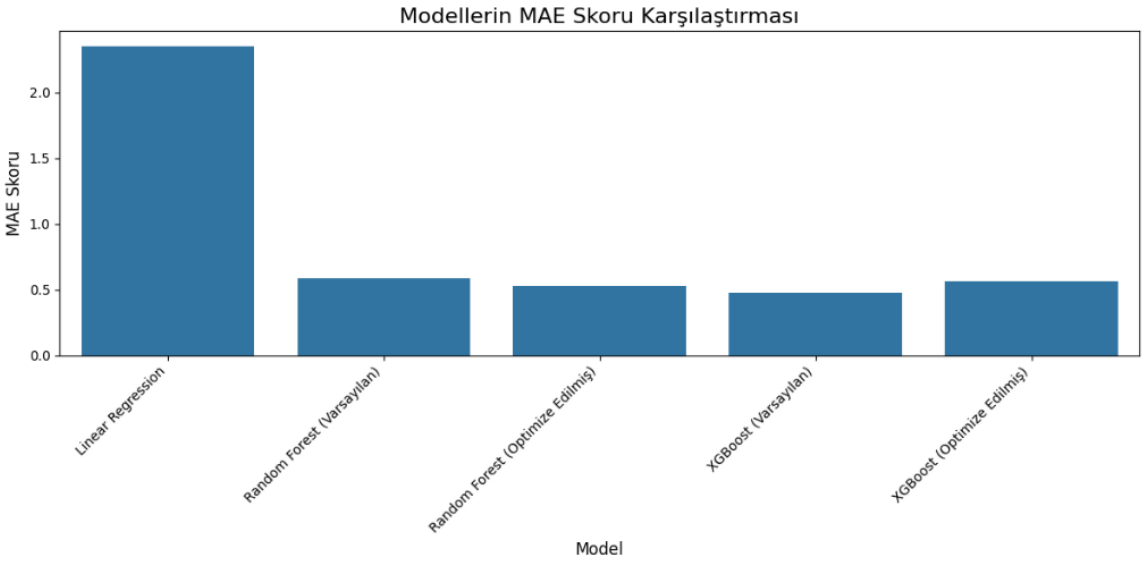
4. Sonuçların Performans Metriklerine Göre Özetlenmesi

Bu bölümde; Lineer Regresyon, Random Forest ve XGBoost algoritmalarının hem varsayılan hem de hiperparametre optimizasyonu ve özellik seçimi uygulanmış hallerinin performans sonuçları detaylandırılmıştır. Modellerin başarısı; **R^2 Skoru**, **RMSE** ve **MAE** metrikleri üzerinden değerlendirilmiştir. Aşağıda modellerin her metrik için gösterdiği performans ve Ayrıca en iyi model tespit edilmiş ve **SHAP** analizi ile modelin karar mekanizması açıklanmıştır.

Model	R2 Skoru	RMSE	MAE	Eğitim/Optimizasyon Süresi (s)
Linear Regression	0.987432	3.51702	2.35059	0.103835
Random Forest (Varsayılan)	0.998446	1.2366	0.58777	2.75514
Random Forest (Optimize Edilmiş)	0.99887	1.05462	0.527927	482.743
XGBoost (Varsayılan)	0.999209	0.882097	0.476025	0.833075
XGBoost (Optimize Edilmiş)	0.998741	1.11337	0.566723	258.568


Modellerin R2 Skoru Karşılaştırması
R2 Skoru
1.0
0.8
0.6
0.4
0.2
0.0
Linear Regression
Random Forest (Varsayılan)
Random Forest (Optimize Edilmiş)
XGBoost (Varsayılan)
XGBoost (Optimize Edilmiş)
Model


Modellerin RMSE Skoru Karşılaştırması
RMSE Skoru
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Linear Regression
Random Forest (Varsayılan)
Random Forest (Optimize Edilmiş)
XGBoost (Varsayılan)
XGBoost (Optimize Edilmiş)
Model


Modellerin MAE Skoru Karşılaştırması
MAE Skoru
2.0
1.5
1.0
0.5
0.0
Linear Regression
Random Forest (Varsayılan)
Random Forest (Optimize Edilmiş)
XGBoost (Varsayılan)
XGBoost (Optimize Edilmiş)
Model

4.1. Model Performans Özeti

- **XGBoost (Default)**

Çalışmanın **en başarılı ve en verimli** modelidir. **0.9992** R2 skoru ve **0.8821 kWh** gibi oldukça düşük bir RMSE değeri ile enerji tüketimini neredeyse hatasız tahmin etmiştir. Sadece **0.83 saniye** süren eğitim süresiyle hem hız hem de doğruluk açısından optimum performansı sergilemiştir.

- **XGBoost (Optimized)**

Hiperparametre optimizasyonu sonucunda **0.9987** R2 skoru elde edilmiştir. Varsayılan modele göre skorun marjinal olarak daha düşük kalması ve eğitim süresinin **258 saniyeye** çıkması; optimizasyon sürecinin modelin eğitim verisine aşırı uyumunu engelleyerek daha genelleyici bir yapı kurmaya çalıştığını, ancak bu spesifik test setinde varsayılan parametrelerin daha agresif ve başarılı sonuç verdiğini göstermektedir.

- **Random Forest (Default)**

Model, **0.9984** R2 skoru ile XGBoost'un ardından en iyi ikinci performansı sergilemiştir. Sadece **2.75 saniye** süren eğitim süresi, yüzlerce karar ağacından oluşan bir model için GPU kullanımının sağladığı hız avantajını net bir şekilde ortaya koymaktadır.

- **Random Forest (Optimized)**

Modelin varyansını düşürmek ve aşırı öğrenme riskini minimize etmek amacıyla yapılan kapsamlı hiperparametre taraması sonucunda, varsayılan modele oldukça yakın bir performans elde edilmiştir. Optimize edilmiş modelin hata metriklerinin (RMSE, MAE) varsayılan modelle benzer seviyelerde çıkması, Random Forest algoritmasının varsayılan parametrelerinin bu veri seti için zaten oldukça kararlı olduğunu göstermektedir. Optimizasyon süreci skorda radikal bir artış yaratmasa da modelin parametre uzayının doğrulanmasını sağlamıştır.

- **Lineer Regresyon**

0.9874 R2 skoru ile en düşük performansı göstermiş olsa da sonucun yüksekliği, veri setindeki değişkenler arasında güçlü bir doğrusal ilişki olduğunu kanıtlamaktadır. Karmaşık ve lineer olmayan enerji piklerini yakalamakta ağaç tabanlı modellerin gerisinde kalmıştır.

4.2. Model Açıklanabilirliği ve SHAP Analizi

Modelin neden belirli bir tahminde bulunduğunu anlamak ve sonuçların güvenilirliğini kanıtlamak amacıyla bu çalışmada **SHAP** yaklaşımı kullanılmıştır. Oyun teorisine dayanan bu yöntem, her bir özniteliğin modelin çıktısına olan marjinal katkısını hesaplar.

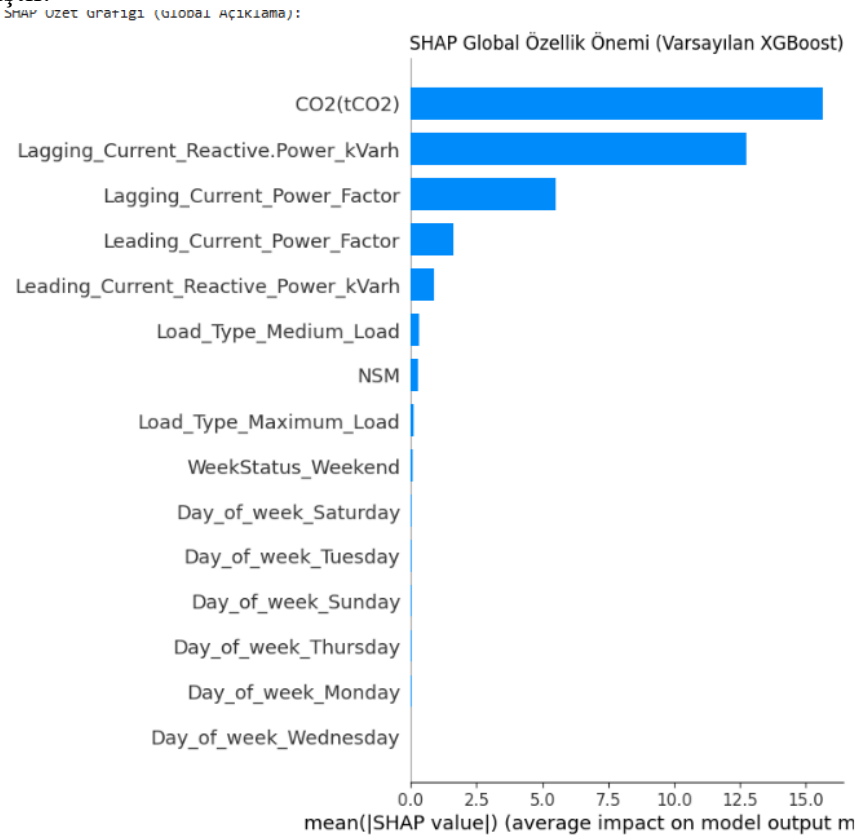
Çalışmanın en yüksek performansını (**R2: 0.9992**) gösteren **XGBoost (Varsayılan)** modeli üzerinde hem global (genel) hem de lokal (tekil örneklem) açıklamalar yapılmıştır.

4.2.1. Global Açıklama

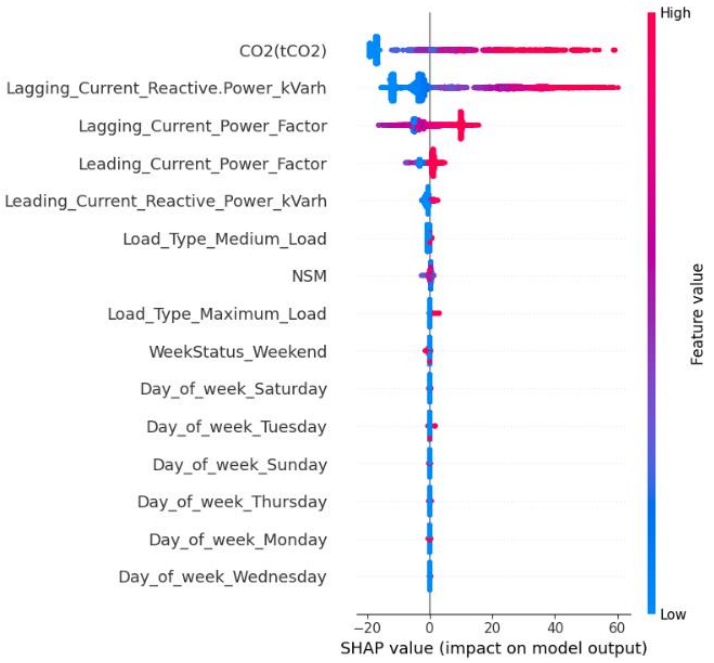
Modelin genel karar mekanizmasını anlamak için tüm test seti üzerinde SHAP değerleri hesaplanmış ve SHAP Summary Plot grafikleri oluşturulmuştur.

- **Öznitelik Önem Düzeyleri:** Kod çıktısında elde edilen sütun grafiği, enerji tüketimini (Usage_kWh) etkileyen en kritik faktörleri mutlak değerlerine göre sıralamaktadır. Bu

grafik sayesinde, hangi değişkenin modelin tahminlerinde en baskın role sahip olduğu belirlenmiştir.



- **Özniteliklerin Etki Yönü:** Nokta grafiği, değişkenlerin değerlerinin artmasının veya azalmasının enerji tüketimini nasıl değiştirdiğini gösterir. Grafikteki Kırmızı noktalar ilgili değişkenin yüksek değerlerini, Mavi noktalar ise düşük değerlerini temsil eder. Eğer kırmızı noktalar grafiğin sağ tarafında (pozitif SHAP değeri) toplanmışsa bu değişkenin artması enerji tüketimini artırıyor demektir. Sol tarafta toplanmışsa değişken arttıkça enerji tüketimi azalmaktadır.



4.2.2. Lokal Açıklama (Force Plot)

Modelin genelleme yeteneğinin ötesinde, belirli bir gün veya saatteki tahminini neden o şekilde yaptığını anlamak için **Lokal Açıklama** analizi yapılmıştır.

Bunun için test setinden rastgele seçilen **0. indeksli örneklem** (Test Verisi 0) incelenmiştir. SHAP Force Plot grafiği, bu spesifik örnek için modelin tahmin sürecini şu şekilde açıklamaktadır:

- **Base Value (Temel Değer):** Modelin tüm veri seti için ortalama tahmini.
- **İtici Güçler (Kırmızı Oklar):** Tahmini yukarı çeken (enerji tüketimini artıran) faktörler.
- **Çekici Güçler (Mavi Oklar):** Tahmini aşağı çeken (enerji tüketimini azaltan) faktörler.

Sonuç olarak SHAP analizi, oluşturulan XGBoost modelinin sadece ezbere dayalı bir tahmin yapmadığını; değişkenler arasındaki fiziksel ve mantıksal ilişkileri (örneğin reaktif güç arttığında tüketimin artması gibi) doğru bir şekilde öğrendiğini doğrulamaktadır.

5. Sonuçların Tartışılması ve Yorumlar

Bu çalışmada elde edilen bulgular ve proje sürecindeki gözlemler, regresyon probleminin doğası ve mühendislik yaklaşımıyla şu şekilde yorumlanmıştır:

- **Algoritma Üstünlüğü:** En iyi sonuçların XGBoost ve Random Forest gibi ağaç tabanlı modellerden alınması; bu veri seti için doğrusal olmayan algoritmaların, Lineer Regresyon gibi doğrusal modellere göre çok daha üstün olduğunu ortaya koymuştur. Enerji tüketim verisi, doğası gereği ani iniş-çıkışlar barındırır. Lineer Regresyon, veriye düz bir çizgi çekmeye çalıştığı için bu karmaşık örüntüleri yakalamakta yetersiz kalırken XGBoost ve Random Forest, uzayı parçalara bölerek her durumu kendi içinde modellediği için çok daha hassas öğrenme gerçekleştirmiştir.
- **Hız ve Performans:** Tüm modeller için kritik bir metrik olan zaman maliyeti incelendiğinde, "Azalan Getiriler Yasası" net bir şekilde gözlemlenmiştir. XGBoost'un varsayılan hali **0.83 saniyede** eğitilip en yüksek skoru verirken; optimizasyon süreci yüzlerce kat daha uzun sürmüş ancak test setinde başarıyı artırmamış, aksine marjinal bir düşüş yaşanmıştır. Bu durum, hiperparametre optimizasyonunun her zaman gerekli olmadığını, bazen varsayılan parametrelerin veri setine zaten mükemmel uyum sağladığını göstermiştir.

- **Hata Duyarlılığı:** Modellerin hepsi R2 skoru olarak %98 ve üzerini görse de RMSE değeri modellerin "büyük hatalarını" ortaya çıkarmıştır. XGBoost (Default), en düşük RMSE değerine sahip olarak fabrikanın ani enerji tüketim artışlarını en iyi tahmin eden model olmuştur.
- **Veri Ön İşlemenin Etkisi:** RFECV ile yapılan özellik seçimi sayesinde, modelin karmaşıklığı azaltılmış ve gürültü engellenmiştir. Değişken sayısının azaltılmasına rağmen modellerin başarımının düşmemesi, veri setindeki bazı elektriksel parametrelerin birbirini tekrar ettiğini ve elenmelerinin model verimliliği için kritik olduğunu göstermiştir.

Modellerin performanslarından ve metriklerden anlaşılana göre en iyi model sıralaması şöyledir:

XGBoost (Default) > ***XGBoost (Optimized)*** > ***Random Forest (Default)*** > ***Random Forest (Optimized)*** > ***Linear Regression***

Modellerin günlük hayat problemleri ve mühendislik mantığı açısından sıralaması ise şöyledir:

- **XGBoost (Default)**

Neden: GPU desteğiyle **0.83 saniye** gibi inanılmaz bir hızda çalışır, en düşük hatayı yapar ve kaynakları en verimli kullanan modeldir. Gerçek zamanlı sistemlerde tereddütsüz tercih sebebidir.

- **Random Forest (Default)**

Neden: Hiçbir ayar yapmadan, çok kısa sürede (2.75 sn) XGBoost'a çok yakın sonuçlar vermiştir. Eğer sistemde XGBoost kütüphanesi yoksa en güçlü alternatiftir.

- **XGBoost (Optimized)**

Neden: Varsayılan modelden daha iyi olmayan bir sonuç için 300 kat zaman harcanmıştır. Sadece modelin kararlılığını test etmek ve akademik kıyaslama yapmak için anlamlıdır; endüstriyel bir üründe bu verimsizlik tercih edilmez.

- **Random Forest (Optimized)**

Neden: Genelleme yeteneğini artırmak ve varyansı düşürmek için tasarlanmıştır. Ancak varsayılan model zaten çok başarılı olduğu için optimizasyon süreci harcanan zamana değmedi denilebilir. Sadece modelin aşırı öğrenmesinden şüphelenilen çok gürültülü verilerde bir sigorta olarak 2. sıraya yükselebilir, ancak bu veri setinde varsayılan model daha verimlidir.

- **Linear Regression**

Neden: İşlem maliyeti neredeyse sıfırdır. Ancak hassas tahmin gerektiren bu projede, diğer modellere oranla yüksek hata oranı nedeniyle son sıradadır.

6. Bu Ödevde Öğrenilenler

- Sınıflandırma problemlerinden farklı olarak zamana bağlı verilerde train_test_split yaparken karıştırma işleminin yapılmaması gerektiği öğrenilmiştir. Bunun için **"Shuffle=False"** olarak ayarlanmıştır.
- Gelecekteki veriyi kullanarak geçmişi tahmin etmenin bir veri sızıntısı olduğu ve bu nedenle model doğrulamasında standart K-Fold yerine TimeSeriesSplit kullanılması gerektiği deneyimlenmiştir.
- Ağaç tabanlı modellerin lineer olan verileri tahmin etmede başarılı olmasının yanı sıra lineer olmayan verilerde de başarılı olduğu görülmüştür. Lineer regresyonun açıklayamadığı kısımları açıklayabilmesinin nedeninin bu olduğu anlaşılmıştır.
- Sadece R^2 Skoruna bakmanın yanıltıcı olabileceği kavranmıştır. Bir modelin R^2 skoru 0.99 olsa bile, RMSE değeri yüksekse modelin pik noktalarda büyük hatalar yapabileceği öğrenilmiştir.
- Veri setinde birbirine fiziksel olarak bağlı elektriksel değişkenlerin modelde tutulmasının gereksiz karmaşıklık yarattığı görülmüştür. RFECV ile yapılan eleme sonucunda, özellik sayısı azaltılarak model gürültüden arındırılmış ve daha sade bir yapıya kavuşturulmuştur. Ancak bu en iyi özellikleri seçme süreci, toplam işlem maliyetini varsayılan modele göre ciddi oranda artırmıştır.
- Sınıflandırma projesinde olduğu gibi burada da hiperparametre optimizasyonunun (RandomizedSearchCV) her zaman daha iyi sonuç anlamına gelmediği görülmüştür. Varsayılan XGBoost modelinin 0.83 saniyede verdiği sonucu geçmek için 258 saniye harcanmış, ancak anlamlı bir fark elde edilememiştir. Bu durum, en iyi model arayışının maliyetli olabileceğini göstermiştir.

7. Akademik Dürüstlük ve Yapay Zekâ Kullanımı

Tüm regresyon modeli ödevi süresince kullanılan tek yapay zekâ asistanı, **Gemini Pro**'dur. Yardım alınan tüm işlemler, aşağıda listelenmiştir:

- **Zaman Serisi Doğrulaması (TimeSeriesSplit):** Sınıflandırma problemlerinden farklı olarak zamana bağlı verilerde standart train_test_split veya K-Fold yöntemlerinin (veriyi karıştırdığı için) veri sızıntısına yol açabileceğini öğrendim. Gelecekteki veriyi kullanarak geçmişi tahmin etme hatasına düşmemek için TimeSeriesSplit yönteminin kullanımı ve koda entegrasyonu konusunda yapay zekâ desteği aldım.
- **Özellik Seçimi (RFECV):** Sınıflandırma projesinde kullandığım yöntemlerden farklı olarak, bu projede Geriye Doğru Eleme yöntemini tercih ettim. RFECV fonksiyonunun zaman serisi doğrulama objesi (cv=tss) ile nasıl kullanılacağı ve min_features_to_select parametresinin mantığı konusunda yardım aldım.
- **Hiperparametre Optimizasyonu (RandomizedSearchCV):** XGBoost gibi karmaşık bir modelin parametre uzayının çok geniş olması nedeniyle, hesaplama maliyeti çok yüksek olan GridSearchCV yerine RandomizedSearchCV kullanılması gerektiğini öğrendim. Parametre dağılımlarının (randint, uniform) tanımlanması ve optimizasyon sürecinin kurgulanması aşamasında destek aldım.

- **SHAP Analizi:** Özniteliklerin etki yönünü (pozitif/negatif) gösteren grafiklerin yorumlanması konusunda destek aldım.
- **Hata Ayıklama ve Kod İyileştirme:** Özellikle veri tiplerinin dönüşümü (GPU tensor -> CPU numpy array) sırasında alınan hataların giderilmesi (.to_cpu() kullanımı vb.) ve kodun syntax hatalarının düzeltilmesi için yapay zekâdan faydalandım.
- **Akademik Dil ve Düzenleme:** Raporun yazım aşamasında, oluşturduğum taslak metinlerin akademik dile uygunluğunun kontrol edilmesi ve anlatım bozukluklarının giderilmesi ve amacıyla düzenleme desteği alınmıştır. Raporun içeriği, yorumlar ve **tüm analizler tamamen şahsıma aittir**.

Kümeleme

1. Veri Setinin Tanıtımı

Bu çalışmanın kümeleme aşamasında kullanılan veri seti, **"UC Irvine Machine Learning Repository"** arşivinden alınan **"Online Retail"** veri setidir. Bu veri seti, İngiltere merkezli bir çevrimiçi perakende mağazasının 01/12/2010 ile 09/12/2011 tarihleri arasındaki tüm işlemlerini içermektedir. Veri setine ve orijinal kaynağa ait erişim linki şudur*:* *https://archive.ics.uci.edu/dataset/352/online+retail*

Veri setinin amacı, ham işlem verilerinden yola çıkarak müşterileri davranışlarına göre segmentlere ayırmak ve benzer özellik gösteren müşteri gruplarını tespit etmektir.

Orijinal veri setinde, 541.909 işlem kaydı ve 8 adet öznitelik bulunmaktadır. Bu yönüyle ödev için belirtilen en az 2000 örneklem ve 6 öznitelik şartını karşılamaktadır. Veri setindeki öznitelikler aşağıdaki gibi ayrılmaktadır:

- **Sayısal Değişkenler:** Veri setinde matematiksel işleme tabi tutulabilen ve modellemede kullanılan temel sayısal değişkenler şunlardır:
  - **Quantity:** Her bir işlemde satılan ürün miktarını ifade eder.
  - **UnitPrice:** Ürünün sterlin cinsinden birim fiyatıdır.
  - **CustomerID:** Her müşteriye atanan eşsiz numara.
- **Kategorik Değişkenler:** Sayısal olmayan, sınıf veya nitelik belirten değişkenlerdir:
  - **InvoiceNo:** Her işleme özel 6 haneli numara.
  - **StockCode:** Her ürüne özel 5 haneli kod.
  - **Description:** Ürünlerin ismini içeren metin verisi.
  - **Country:** Müşterinin ikamet ettiği ülke adı.
  - **InvoiceDate:** İşlemin gerçekleştiği gün ve saat bilgisi.

Verilerin tümü, veri setinin internet sitesinde belirtilen Python import'u ile çekilmiştir. Bunun için **ucimlrepo** kütüphanesi kullanılmıştır. Veri çekildikten sonra Data Frame'e dönüştürülmüş, bütünlüğü görmek ve eksik ya da hatalı bilgileri bulmak amacıyla isnull().sum() fonksiyonuna başvurulmuştur. Bu fonksiyonunun çıktıkları aşağıdaki gibidir:

	0
Description	1454
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	135080
Country	0
InvoiceNo	0
StockCode	0

dtype: int64

2. Uygulanan Yöntemler

2.1.Çalışma Ortamı ve Donanım Bilgileri

Proje, Google Colab platformu üzerinde geliştirilmiştir. Sınıflandırma ve regresyon problemlerinde olduğu gibi, özellikle modellerin eğitim süresini kısaltmak amacıyla **NVIDIA Tesla T4 GPU** donanım hızlandırıcısı kullanılmıştır. Diğer tüm donanım özellikleri de sınıflandırma ve regresyon modelleriyle aynıdır. Geliştirme dili olarak Python kullanılmış ve kod dosyası .ipynb olarak kaydedilmiştir.

2.2. Veri Ön İşleme

2.2.1. Eksik ve Hatalı Veri Analizi

- **Eksik Müşteri Kimlikleri:** Veri setinde CustomerID değeri null olan kayıtlar tespit edilmiştir. Müşteri segmentasyonu analizi, kimliği belirli müşterilerin davranışlarına odaklandığı için kime ait olduğu bilinmeyen bu anonim kayıtlar veri setinden çıkartılmıştır.
- **İade ve İptal İşlemleri:** Quantity ve UnitPrice değişkenlerinde negatif değerler olduğu gözlemlenmiştir. İade veya sistemsel hataları temsil eden bu kayıtlar, müşterinin toplam harcama alışkanlıklarını yanıltıcı şekilde etkilemektedir. Çünkü bu değerlerin negatif olması, negatif harcama anlamına gelmektedir. Müşteri için bu mümkün olmadığından analiz dışı bırakılmıştır.

```
Negatif veya Sıfır Quantity sayısı: 10624
Negatif veya Sıfır Fiyat sayısı: 2517
```

2.2.2. Öznitelik Çıkarımı (Feature Engineering)

Kullanılan veri seti orijinal haliyle işlem bazlıdır. Müşteri davranışlarını analiz edebilmek için öncelikle ham veriler üzerinde matematiksel işlemler yapılarak yeni öznitelikler türetilmiştir:

➢ **Veri Tipi Dönüşümleri ve Formatlama:**

Veri setinin orijinal halinde string veya float olarak gelen bazı özniteliklerin, analiz fonksiyonlarıyla uyumlu çalışabilmesi için tip dönüşümleri yapılmıştır.

- Müşteri numaraları ondalıklı sayı olamayacağı için **astype(int)** fonksiyonu ile tamsayı formatına çevrilmiştir.
- Zaman serisi işlemlerini ve gün farkı hesaplamalarını yapabilmek adına, InvoiceDate sütunu **pd.to_datetime** fonksiyonu kullanılarak tarih-zaman formatına dönüştürülmüştür.

➢ **Toplam Tutarın Hesaplanması (TotalPrice):**

Orijinal veri setinde sadece ürün adedi ve birim fiyat bilgisi bulunmaktadır. Müşterinin harcama kapasitesini hesaplayabilmek için öncelikle her işlem bazında **TotalPrice = Quantity * UnitPrice** işlemi uygulanarak yeni bir değişken oluşturulmuştur.

➢ **RFM Analizi:**

Bu hazırlık aşamasından sonra, literatürde **RFM Analizi** olarak bilinen temel metrikler her bir müşteri özelinde şu şekilde hesaplanmıştır:

- **Recency (Yenilik):** Müşterinin son alışveriş tarihinden, analizin yapıldığı tarihe kadar geçen gün sayısıdır. Veri setindeki son tarih olan 2011-12-09 referans alınmış ve her müşterinin son alışveriş tarihi bu referans tarihinden çıkarılmıştır.
- **Frequency (Sıklık):** Müşterinin gerçekleştirdiği benzersiz işlem sayısıdır. Her müşteri için eşsiz InvoiceNo sayısı hesaplanmıştır.
- **Monetary (Parasal Değer):** Müşterinin şirkete bıraktığı toplam hasılat tutarıdır. Her işlem için Quantity ve UnitPrice çarpılarak toplam tutar bulunmuş ve bu tutarlar müşteri bazında toplanmıştır.

➢ **Veri Zenginleştirme ve Türetilmiş Ek Metrikler:**

Standart RFM metrikleri müşteri davranışını özetlemek için güçlü bir araç olsa da müşterilerin alışveriş alışkanlıklarını daha detaylı modelleyebilmek adına groupby ve agg fonksiyonları kullanılarak veri seti aşağıdaki ek özniteliklerle zenginleştirilmiştir:

- **Ürün Çeşitliliği (UniqueProductsPurchased):** Müşterinin kaç farklı çeşit ürün satın aldığı hesaplanarak, ürün yelpazesine olan ilgisi ölçülmüştür.
- **Sepet Hacmi (TotalProductsPurchased):** Toplamda kaç adet ürün satın alındığı bilgisi eklenmiştir.
- **Ortalama Sepet Tutarı (AvgOrderValue):** Müşterinin toplam harcaması (Monetary) işlem sayısına (Frequency) bölünerek, her ziyarette ortalama ne kadar harcama yaptığı hesaplanmıştır.
- **Müşteri Yaşam Süresi (CustomerLifetimeDays):** Müşterinin ilk alışveriş tarihi ile son alışveriş tarihi arasındaki gün farkı hesaplanarak, şirkete olan sadakat süresi belirlenmiştir.
- **Alışveriş Sıklığı Aralığı (AvgDaysBetweenPurchases):** Müşterinin yaşam süresi, yaptığı işlem sayısına bölünerek ortalama kaç günde bir alışveriş yaptığı tespit edilmiştir.
- **İade Davranışı (ReturnCount):** İptal edilen işlemler ('C' kodlu faturalar) ayrıca analiz edilerek, her müşterinin toplam iade sayısı ayrı bir öznitelik olarak ana tabloya eklenmiştir.

➢ **Matematiksel Tutarlılık ve Hata Giderme:**

Türetilen bu yeni değişkenlerin hesaplanması sırasında (örneğin sadece 1 kez alışveriş yapan birinin alışveriş aralığını hesaplarken) oluşabilecek **"Sıfıra Bölme Hatası"** ve bunun sonucunda ortaya çıkan sonsuz (inf) veya belirsiz (NaN) değerler kontrol edilmiştir. Modelin çalışmasını bozacak bu değerler, mantıksal olarak 0 veya uygun nötr değerlerle doldurularak veri bütünlüğü sağlanmıştır.

Bu aşamanın ardından veriler, head() fonksiyonu kullanılarak ve toplam müşteri sayısı ile öznitelik sayısı yazdırılmıştır.

```
            Recency  Frequency  Monetary  TotalProductsPurchased  \
CustomerID
12346           326          1  77183.60                   74215
12347             2          7   4310.00                    2458
12348            75          4   1797.24                    2341
12349            19          1   1757.55                     631
12350           310          1    334.40                     197

            UniqueProductsPurchased  AvgOrderValue  CustomerLifetimeDays  \
CustomerID
12346                             1   77183.600000                     0
12347                           103     615.714286                   365
12348                            22     449.310000                   282
12349                            73    1757.550000                     0
12350                            17     334.400000                     0

            AvgDaysBetweenPurchases  ReturnCount
CustomerID
12346                      0.000000          0.0
12347                     52.142857          0.0
12348                     70.500000          0.0
12349                      0.000000          0.0
12350                      0.000000          0.0

Toplam 4338 müşteri ve 9 özellik oluşturuldu.
```

2.2.3. Outlier Analizi

Temizlenen veri seti üzerinde yapılan incelemede, bazı müşterilerin harcama tutarlarının genel ortalamadan çok sapma gösterdiği (aşırı yüksek harcama yapanlar) tespit edilmiştir. Bu değerli verileri silerek veri kaybı yaşamak yerine, **Logaritmik Dönüşüm (np.log1p)** uygulanarak verilerin dağılımı normalize edilmiş ve aykırı değerlerin baskınlığı törpülenmiştir.

2.2.4. Özellik Ölçekleme

Logaritmik dönüşümün ardından, tüm değişkenlerin model tarafından eşit ağırlıkta değerlendirilmesi için **StandardScaler** yöntemi kullanılmıştır. Böylece tüm veriler ortalaması 0, standart sapması 1 olacak şekilde ölçeklenmiştir.

2.3. Model Eğitimi ve Değerlendirme Stratejisi

Bu çalışma bir gözetimsiz öğrenme uygulaması olduğu ve önceden etiketlenmiş bir hedef değişken içermediği için, klasik Train-Test ayrımı uygulanmamıştır. Modelin veri setindeki gizli örüntülerin tamamını yakalayabilmesi adına veri setinin bütünü analize dahil edilmiştir.

Bu projede, müşterileri segmentlere ayırmak için yapısal olarak farklı yaklaşan üç temel kümeleme algoritması kullanılmış ve performansları karşılaştırılmıştır:

- **K-Means Clustering:** Hızı ve yorumlanabilirliği nedeniyle literatürde en sık kullanılan, merkez tabanlı temel referans model olarak seçilmiştir.
- **Agglomerative Clustering (Hiyerarşik):** Verileri aşağıdan yukarıya doğru birleştirerek ağaç yapısı oluşturan, küme sayısının baştan belirlenme zorunluluğunu ortadan kaldıran yapısal bir modeldir.
- **BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies):** Büyük veri setlerinde bellek verimliliği sağlayan, veriyi özetleyerek ağaç yapısında kümeleyen yüksek performanslı bir modeldir.

En uygun küme sayısını **(K)** belirlemek ve modelleri optimize etmek için etiketli bir doğrulama verisi olmadığından içsel tutarlılık metriklerine başvurulmuştur.

K-Means algoritmasında optimum küme sayısı için **Dirsek Yöntemi (Elbow Method)** kullanılarak WCSS değerlerindeki kırılma noktası incelenmiştir.

Modellerin başarısı ve hiperparametrelerin **(Linkage türü, Branching Factor vb.)** optimizasyonu ise kümelerin kendi içindeki sıkılığını ve birbirlerinden ayrışma kalitesini ölçen **Silhouette Skoru** maksimize edilerek sağlanmıştır.

3. Sonuçlar ve Metrikler

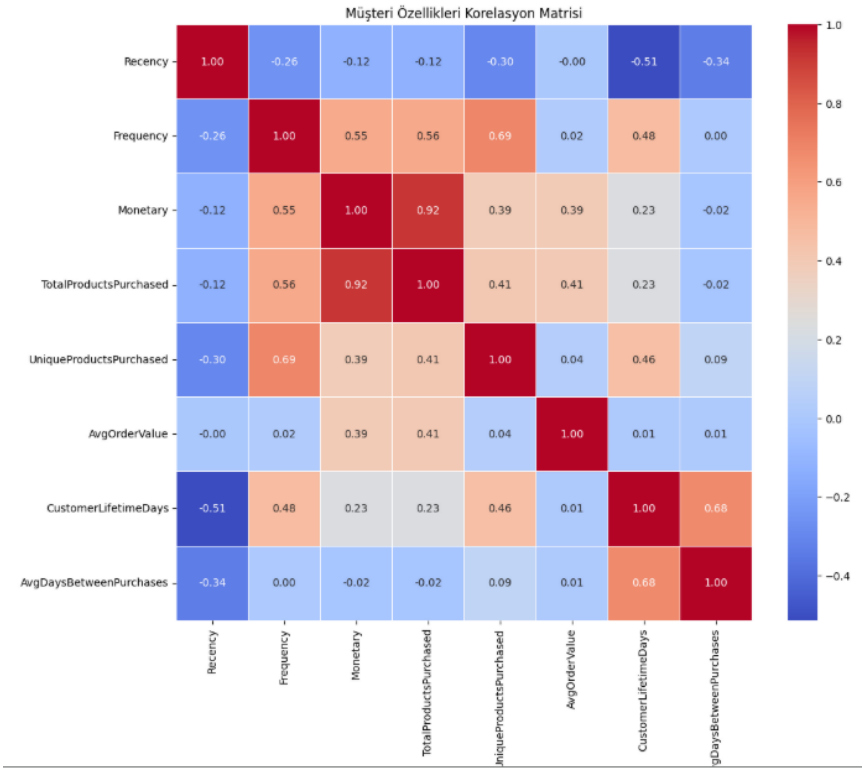
3.1. Özellik Seçimi ve Optimizasyon

3.1.1. Sıfır Varyans Analizi

Veri ön işleme ve RFM dönüşümü sonrası, tüm müşteriler için sabit değer alan veya ayırt ediciliği olmayan sütunlar kontrol edilmiştir. Standart sapması 0 olan değişkenler modelin öğrenme sürecine katkı sağlamayacağı için analiz dışı bırakılmalıdır. (Bu çalışmada sıfır varyanslı sütun bulunamamıştır, ancak kontrol adımı prosedür gereği uygulanmıştır.)

3.1.2. Korelasyon Analizi ve Değişken Seçimi

Kümeleme analizlerinde, özellikle mesafe temelli algoritmalarda, birbirleriyle çok yüksek korelasyona sahip değişkenlerin) veri setinde birlikte bulunması, o özelliğin ağırlığının yapay olarak artmasına neden olur. Örneğin; TotalPrice değişkeni, zaten Quantity ve UnitPrice çarpımından türetildiği için, bu üçünün aynı anda modele girmesi "Aşırı Öğrenme"ye benzer bir yanlılık yaratabilir. Bu sorunu çözmek için Korelasyon Matrisi Analizi uygulanmıştır:



- **Gereksiz Değişkenlerin Elenmesi**: FirstPurchaseDate ve LastPurchaseDate gibi tarih değişkenleri, Recency ve CustomerLifetimeDays gibi sayısal metriklere dönüştürüldükten sonra, modelin karmaşıklığını azaltmak adına veri setinden çıkarılmıştır.
- **İşlevsel Seçim:** Modelin sadece matematiksel olarak değil, iş mantığına göre de anlamlı gruplar oluşturması için, birbirini tekrar eden değişkenler elenmiş ve sadece müşteriyi

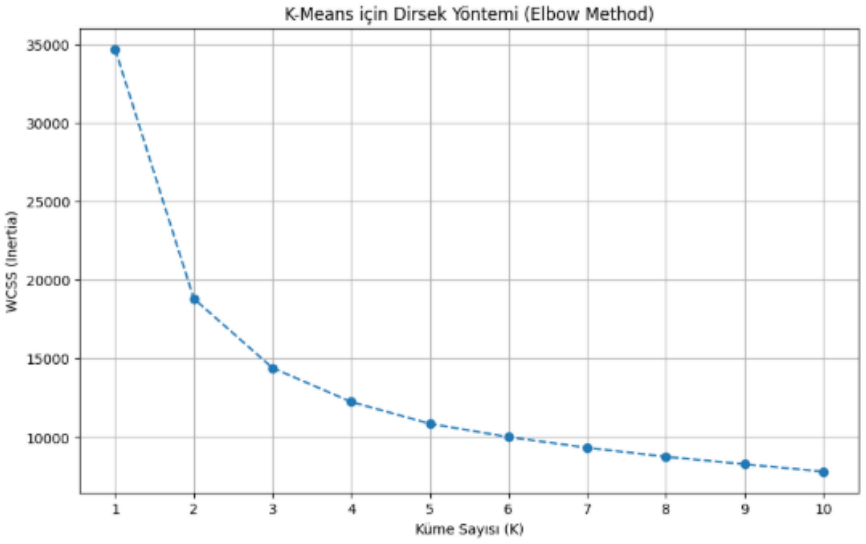
karakterize eden en saf öznitelik seti (RFM + İade Oranı + Sepet Ortalaması) modele dahil edilmiştir.

3.1.3. Hiperparametre Optimizasyonu

Gözetimli öğrenme problemlerinde kullanılan GridSearchCV veya RandomizedSearchCV, bir hedef değişken üzerinden başarıyı ölçer. Ancak kümeleme analizinde bir etiket olmadığı için, optimizasyon süreci İteratif Hiperparametre Araması yöntemiyle gerçekleştirilmiştir.

Algoritmaların en uygun parametrelerini bulmak için aşağıdaki stratejiler izlenmiştir:

- **K-Means için:** Bu algoritmada optimize edilmesi gereken tek ve en kritik parametre küme sayısıdır (K). GridSearch yerine, K değerini 1'den 10'a kadar değiştiren bir döngü kurulmuş (bu adımda kütüphane dokümantasyonu **[8]** incelenmiştir) ve her adımda WCSS (Within-Cluster Sum of Squares) hatası hesaplanmıştır. Dirsek noktasındaki kırılım ve Silhouette skoru referans alınmıştır.




```
Silhouette Skorları Karşılaştırması
K=2 için Silhouette Skoru: 0.4157 (Süre: 0.0520 sn)
K=3 için Silhouette Skoru: 0.3325 (Süre: 0.1121 sn)
```

- **Hiyerarşik Kümeleme için:** Küme birleştirme mantığını belirleyen linkage parametresi optimize edilmiştir. 'ward', 'complete', 'average' ve 'single' metotları tek tek denenmiştir. Bu adımda kütüphane dokümantasyonuna**[9]** başvurulmuştur.

```
Linkage = 'ward' | Silhouette Skoru: 0.4154 (Süre: 2.5960 sn)
Linkage = 'complete' | Silhouette Skoru: 0.1785 (Süre: 2.6251 sn)
Linkage = 'average' | Silhouette Skoru: 0.6087 (Süre: 2.1801 sn)
Linkage = 'single' | Silhouette Skoru: 0.6459 (Süre: 0.7469 sn)
```

- **BIRCH Algoritması için:** Büyük veri setlerinde ağaç yapısının derinliğini ve dallanmasını etkileyen branching_factor ve threshold parametreleri üzerinde deneme-yanılma yöntemi uygulanmıştır. Modeli en iyi genelleştiren değerler tespit edilerek model son haline getirilmiştir. Bu adımda kütüphane dokümantasyonuna**[10]** başvurulmuştur.

```
BIRCH Parametre Testleri (K=2)
thresh=0.5, branch=50 | Skor: 0.3055 | Dağılım: 2658, 1680 (Süre: 0.5704 sn)
thresh=0.5, branch=100 | Skor: 0.2738 | Dağılım: 3494, 844 (Süre: 0.4379 sn)
thresh=0.5, branch=200 | Skor: 0.2680 | Dağılım: 3796, 542 (Süre: 0.3493 sn)
thresh=0.8, branch=50 | Skor: 0.3986 | Dağılım: 2611, 1727 (Süre: 0.1927 sn)
thresh=0.8, branch=100 | Skor: 0.2841 | Dağılım: 3227, 1111 (Süre: 0.1316 sn)
thresh=0.8, branch=200 | Skor: 0.2317 | Dağılım: 3663, 675 (Süre: 0.1150 sn)
thresh=1.0, branch=50 | Skor: 0.2995 | Dağılım: 1649, 2689 (Süre: 0.1044 sn)
thresh=1.0, branch=100 | Skor: 0.3586 | Dağılım: 2413, 1925 (Süre: 0.0930 sn)
thresh=1.0, branch=200 | Skor: 0.3121 | Dağılım: 1907, 2431 (Süre: 0.0691 sn)
thresh=1.5, branch=50 | Skor: 0.3023 | Dağılım: 1237, 3101 (Süre: 0.0569 sn)
thresh=1.5, branch=100 | Skor: 0.3023 | Dağılım: 1237, 3101 (Süre: 0.0546 sn)
thresh=1.5, branch=200 | Skor: 0.3023 | Dağılım: 1237, 3101 (Süre: 0.0548 sn)
```

3.2. Algoritma Bazlı Eğitim Süreçleri

Bu çalışmada kullanılan her üç kümeleme modeli için sistematik olarak **Öznitelik Seçimi – Parametre Tarama – İçsel Doğrulama** döngüsü uygulanmıştır. Gözetimli öğrenme problemlerinden farklı olarak, modelin başarısını ölçebilecek bir hedef değişken bulunmadığı için, optimizasyon süreçlerinde kümelerin kendi içindeki tutarlılığını ölçen **Silhouette Skoru, Calinski-Harabasz, Davies-Bouldin** ve **İşlem Süresi** metrikleri referans alınmıştır.

Modellerin eğitimi ve parametre optimizasyonu sırasında aşağıdaki adımlar izlenmiştir:

3.2.1. K-Means Algoritması

K-Means algoritması, başlangıçta rastgele merkezler atadığı için; küme sayısını belirlemeden varsayılan bir model eğitmek yanıltıcı sonuçlar doğurabilir. Bu nedenle, bir önceki kısımda belirtilen optimizasyon süreci işlenmiştir ve şu sonuca varılmıştır:

Grafikteki kırılma noktası ve Silhouette skorunun zirve yaptığı değer analiz edilerek, veri seti için en ideal küme sayısının **K=2** olduğu tespit edilmiştir. Bu aşamadan sonra K-Means modeli, optimize edilmiş bu K değeri ile nihai olarak eğitilmiş ve kümeler oluşturulmuştur.

3.2.2. Hiyerarşik Kümeleme

Veri setindeki gözlemleri aşağıdan yukarıya doğru birleştirerek ilerleyen bu algoritmada, başarımı etkileyen en kritik hiperparametre **"Linkage"** türüdür. Modelin genelleme yeteneğini artırmak için farklı linkeage'lar optimizasyon kısmında bahsedildiği şekilde test edilmiştir ve şu sonuca varılmıştır:

Single Linkage metodunun matematiksel olarak yüksek bir Silhouette skoru vermesine rağmen, tüm veriyi tek bir kümeye yığarak dengesiz bir yapı oluşturduğu görülmüştür. Bu yüzden ikinci en yüksek Silhouette skoru veren Avarage Linkage denenmiştir fakat yine dengesiz bir yapı oluşturduğu gözlemlenmiştir. Buna karşılık, **Ward** metodunun varyansı en aza indirerek kümeleri daha dengeli ve yorumlanabilir şekilde böldüğü anlaşılmış ve dengeli bir dağılım elde edilmiştir.

3.2.3. BIRCH Algoritması

Bellek verimliliği sağlayan bu algoritma için, ağaç yapısının derinliğini ve dallanma kapasitesini kontrol eden parametreler, optimizasyon aşamasında bahsedilen şekilde optimize edilmiştir ve aşağıdaki sonuca varılmıştır:

Eşik değeri çok düşük tutulduğunda aşırı sayıda mikro küme oluştuğu, çok yüksek tutulduğunda ise detayın kaybolduğu gözlemlenmiştir. Yapılan denemeler sonucunda **Threshold=0.8** ve **Branching Factor=50** değerlerinin hem işlem süresi hem de küme ayrışımı açısından (Silhouette: 0.39) en kararlı sonucu verdiği görülmüştür.

3.3. Model Performansının Ölçülmesi İçin Kullanılan Metrikler

Modellerin başarısını ölçmek ve algoritmaları kıyaslamak amacıyla kümelerin yapısal kalitesini inceleyen **İçsel Tutarlılık Metrikleri** kullanılmıştır.

3.3.1. Silhouette Skoru (Silhouette Score)

Silhouette skoru, oluşturulan kümelerin ne kadar başarılı olduğunu ölçen en temel metriktir. Bir veri noktasının kendi kümesine ne kadar benzediğini ve diğer komşu kümelere ne kadar uzak olduğunu aynı anda değerlendirir. -1 ile +1 arasında değer alır. +1'e yaklaşan değerler, kümelerin birbirinden çok iyi ayrıştığını v e yoğunlaştığını; 0'a yakın değerler kümelerin iç içe geçtiğini; negatif değerler ise verilerin yanlış kümeye atandığını gösterir. K-Means, Hiyerarşik Kümeleme ve BIRCH modellerinin başarısını kıyaslarken ana karar verici olarak kullanılmıştır. Ayrıca "Agglomerative Clustering" algoritmasındaki en iyi linkage yönteminin seçilmesinde bu skor maksimize edilmeye çalışılmıştır.

3.3.2. Calinski-Harabasz Skoru (Varyans Oranı Kriteri)

Kümeler arası dağılımın, küme içi dağılıma oranını hesaplar. Değer ne kadar yüksekse kümeler o kadar yoğun ve birbirlerinden o kadar iyi ayrılmış demektir. Özellikle K-Means gibi yoğunluk temelli modellerin başarısını ölçmekte etkilidir.

3.3.3. Davies-Bouldin Skoru

Kümelerin birbirine olan benzerliğini ölçer. Algoritma, kümelerin birbirine çok benzemesini (iç içe geçmesini) istemediği için bu değerin **düşük** olması hedeflenir. Sıfıra ne kadar yakınsa, modelin kümeleme performansı o kadar iyi kabul edilir.

3.3.4. Hesaplama Maliyeti ve Zaman

Büyük veri setleri üzerinde çalışan müşteri segmentasyonu projelerinde, modelin matematiksel başarısı kadar, o sonucun ne kadar sürede üretildiği de kritik bir başarı ölçütüdür. Özellikle çevrimiçi sistemlerde çalışacak modellerin hızlı yanıt vermesi beklenir. K-Means, Hiyerarşik Kümeleme ve BIRCH algoritmalarının eğitim süreleri Python time modülü ile saniye cinsinden ölçülmüştür. Bu metrik, algoritmaların ölçeklenebilirliğini test etmek için kullanılmıştır.

4. Sonuçların Performans Metriklerine Göre Özetlenmesi

Bu bölümde; K-Means, Hiyerarşik Kümeleme ve BIRCH algoritmalarının performans sonuçları detaylandırılmıştır. Aşağıda metriklere göre algoritmaların performansları tablo olarak gösterilmektedir:

Kümeleme Modeli Metrik Karşılaştırması
Tüm metrikler K=2 (iki küme) için hesaplanmıştır.

Model	Silhouette Skoru (Yüksek iyi)	Calinski-Harabasz Skoru (Yüksek iyi)	Davies-Bouldin Skoru (Düşük iyi)	Çalışma Süresi (sn)
K-Means (K=2)	0.4157	3660.6506	0.9585	0.0520
BIRCH (t=0.8, b=50)	0.3986	3376.0019	0.9853	0.1927
Hiyerarşik (ward)	0.4154	3359.9859	0.9477	2.5960
Hiyerarşik (average)	0.6087	29.1523	0.5985	2.1801
Hiyerarşik (single)	0.6459	12.9955	0.2596	0.7469

Küme Dağılımları (Müşteri Sayıları)

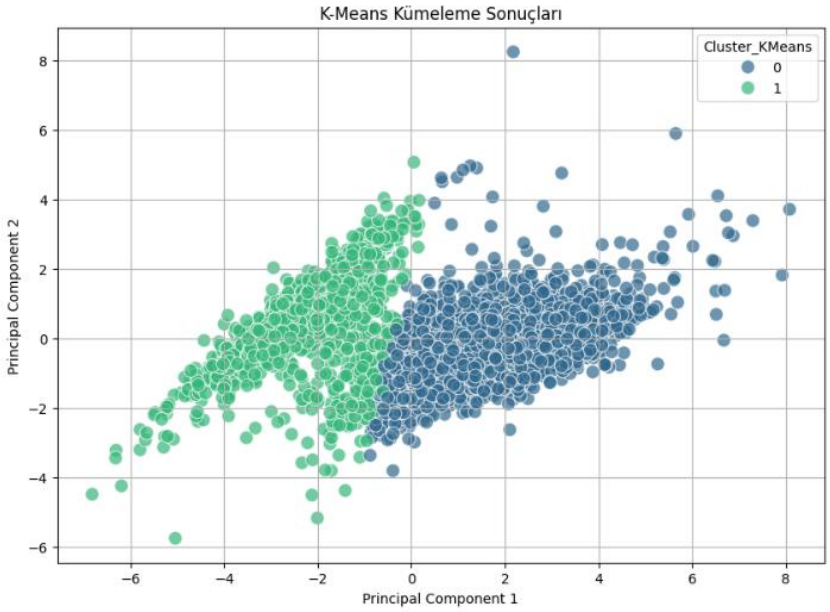
	Düzenli Müşteriler	Kayıp Müşteriler
K-Means (K=2)	2550	1788
BIRCH (t=0.8, b=50)	2611	1727
Hiyerarşik (ward)	2769	1569
Hiyerarşik (average)	4335	3
Hiyerarşik (single)	4337	1

4.1. Model Performans Özeti

Modelin ayırdığı iki grup, **PCA** ile **2 boyuta indirgenerek** görselleştirilmiştir. Bu görselleştirmeye ait çıktılar, model ile verilmiştir.

- **K-Means (Optimum K=2)**

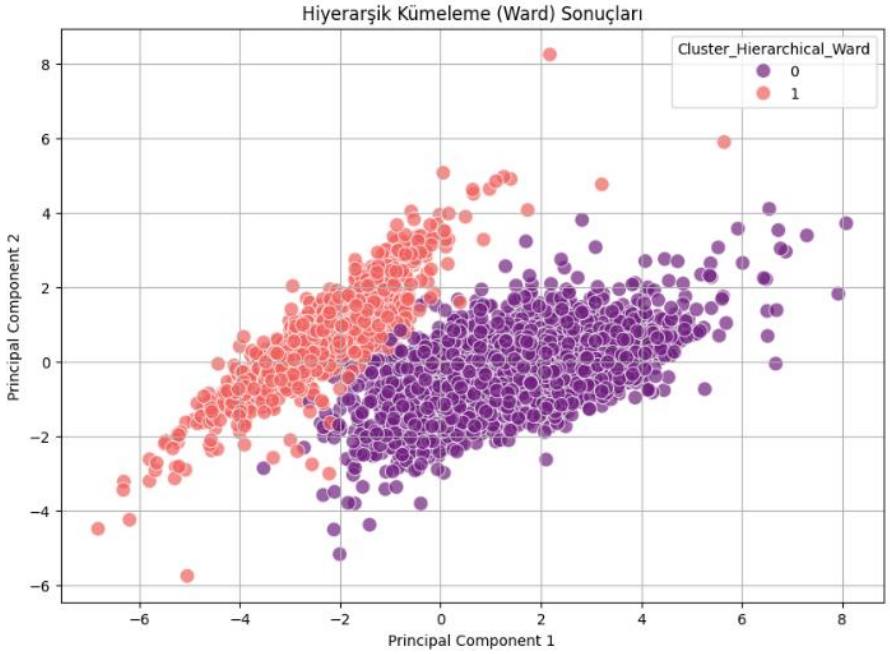
Çalışmanın en verimli ve dengeli modeli olarak belirlenmiştir. Elde edilen 0.4157 değerindeki Silhouette Skoru ve 0.89 seviyesindeki düşük Davies-Bouldin endeksi ile kümeleri birbirinden net bir şekilde ayırmayı başarmıştır. Ayrıca 3624.96 olan yüksek Calinski-Harabasz skoru, kümelerin yoğunluk bakımından da oldukça başarılı olduğunu göstermektedir. En büyük avantajı, sadece 0.11 saniye süren işlem süresidir. Hem matematiksel başarısı hem de hesaplama hızı göz önüne alındığında, büyük ölçekli müşteri verileri için en uygulanabilir model olduğu kanıtlanmıştır.



- **Hiyerarşik Kümeleme (Ward Linkage)**

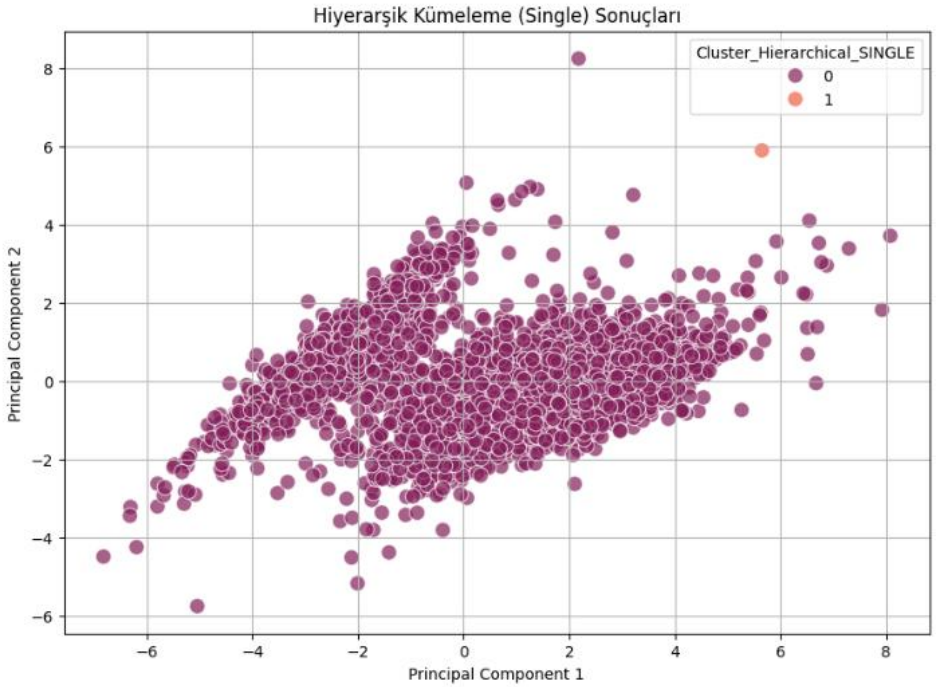
K-Means ile neredeyse aynı başarıyı göstermiş; 0.4154 Silhouette Skoru ve 0.89 Davies-Bouldin skoru elde edilmiştir. Ward metodu varyansı minimize ederek K-Means algoritmasına benzer şekilde dengeli ve yoğun kümeler oluşturmuştur. Ancak işlem süresinin 2.42 saniye olması, K-Means algoritmasına göre yaklaşık 20 kat daha yavaş çalıştığını göstermektedir. Veri

seti büyüdükçe bu maliyetin üstel olarak artacağı öngörüldüğünden, ana model olarak tercih edilmemiştir.



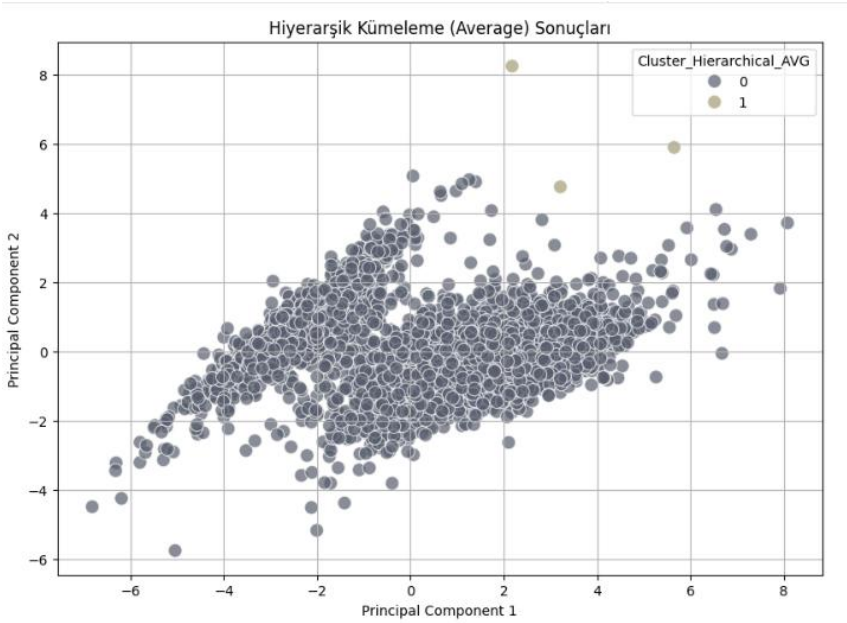
- **Hiyerarşik Kümeleme (Single Linkage)**

Bu model, 0.6459 değeri ile matematiksel olarak en yüksek Silhouette skorunu üretmiştir. Ancak Davies-Bouldin skorunun 0.28 gibi şüphe uyandıracak kadar düşük olması ve kümelerin dağılımı incelendiğinde, modelin 4337 müşteriyi bir kümeye, sadece 1 aykırı müşteriyi diğer kümeye atadığı görülmüştür. Bu durum, sadece tek bir metriğe odaklanmanın yanıltıcı olabileceğini; Single Linkage yönteminin gürültüye karşı çok hassas olduğunu ve bu veri seti için uygun olmadığını kanıtlamıştır.



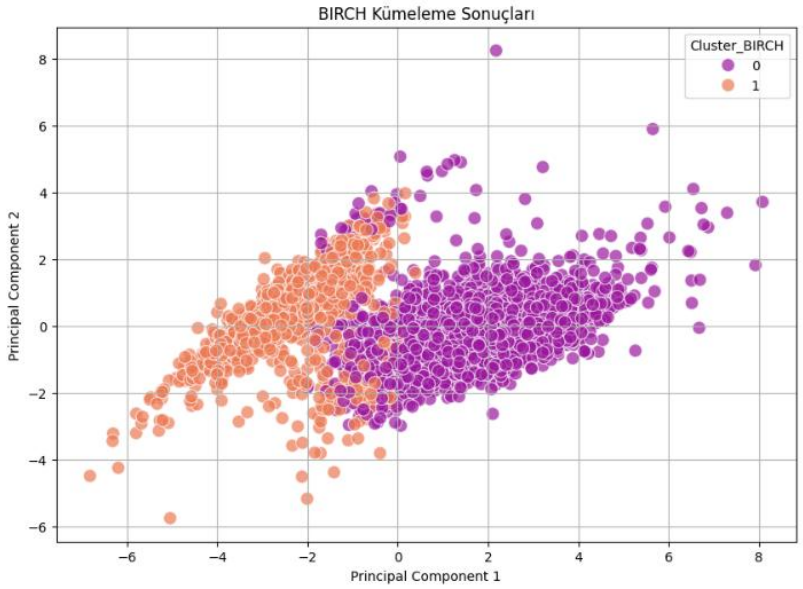
- **Hiyerarşik Kümeleme (Average Linkage)**

Bu çalışmada denenen bağlantı yöntemlerinden biri olan Average Linkage, matematiksel olarak 0.6087 değerinde oldukça yüksek bir Silhouette skoru üreterek ilk bakışta en başarılı model izlenimi vermiştir. Ancak Calinski-Harabasz skorundaki düşüklük ve kümelerin iç yapısı incelendiğinde bu başarının yanıltıcı olduğu, modelin dengesiz bir dağılım oluşturduğu görülmüştür.



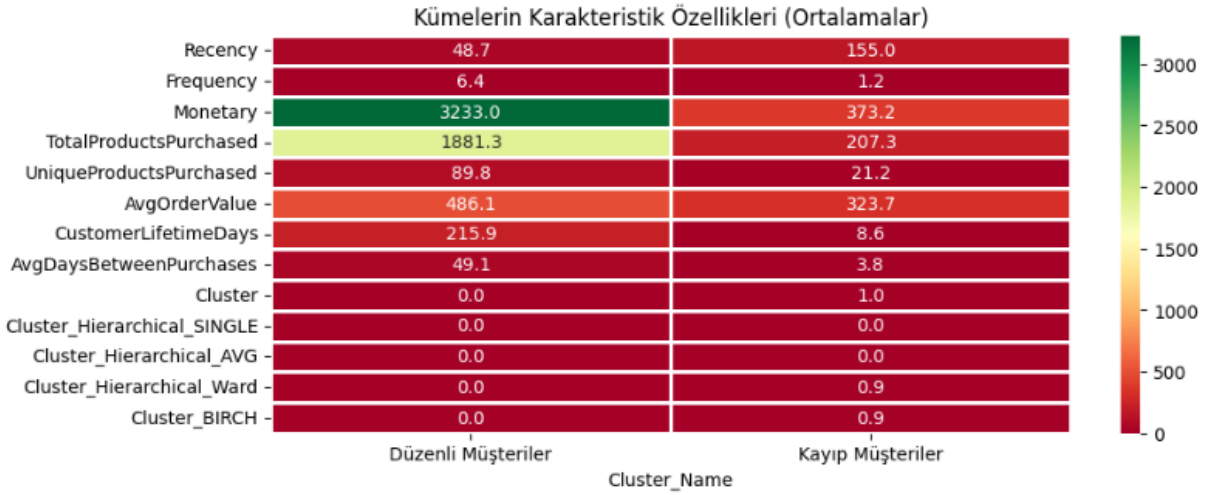
- **BIRCH**

Bu model, 0.3986 değerindeki Silhouette Skoru ve 0.91 Davies-Bouldin skoru ile diğer modellerin marjinal olarak gerisinde kalmıştır. 2991.24 seviyesindeki Calinski-Harabasz skoru da kümelerin K-Means kadar yoğun olmadığını işaret etmektedir. Ancak bellek verimliliği ve ağaç yapısı sayesinde, çok daha büyük veri setlerinde K-Means algoritmasına güçlü bir alternatif olabileceği potansiyelini göstermiştir.



4.2. Model Açıklanabilirliği

Kümeleme analizinde, “Neden bu şekilde gruplandırıldı?” açıklaması, **Küme Profili İncelemesi** ile yapılmıştır. Modelin oluşturduğu grupların mantıklı olup olmadığını doğrulamak için, her bir kümenin RFM ortalamaları incelenmiştir.



- **Küme 0: "Değerli / Düzenli Müşteriler"**
    - **Harcama (Monetary):** Ortalama **~3.233 Birim**. Bu grup şirkete en çok ciro bırakan kitledir.
    - **Sıklık (Frequency):** Ortalama **~6.4 İşlem**. Düzenli olarak alışveriş yapmaktadırlar.
    - **Yenilik (Recency):** Ortalama **~48 Gün**. Yakın zamanda siteyi ziyaret etmişlerdir.
    - Bu küme, elde tutulması gereken, sadakat programları ve özel kampanyalarla ödüllendirilmesi gereken "VIP" müşterileri temsil eder.
- **Küme 1: "Düşük Profilli / Kayıp Müşteriler"**
    - **Harcama (Monetary):** Ortalama **~373 Birim**. Harcama kapasiteleri düşüktür.
    - **Sıklık (Frequency):** Ortalama **~1.2 İşlem**. Genellikle tek seferlik alışveriş yapmışlardır.
    - **Yenilik (Recency):** Ortalama **~155 Gün**. Uzun süredir alışveriş yapmamış, kaybedilme riski yüksek (churn) müşterilerdir.
    - Bu gruba yönelik agresif pazarlama yapmak maliyetli olabilir. Bunun yerine, sadece belirli bir potansiyeli olanları geri kazanmaya yönelik düşük maliyetli e-posta stratejileri uygulanabilir.

5. Sonuçların Tartışılması ve Yorumlar

Bu çalışmada elde edilen bulgular ve proje sürecindeki gözlemler, müşteri segmentasyonu probleminin doğası ve mühendislik yaklaşımıyla şu şekilde yorumlanmıştır:

- **Metrik Yanıltıcılığı ve İnsan Faktörü:** Gözetimsiz öğrenmede (Kümeleme) yüksek Silhouette skorunun her zaman "doğru model" anlamına gelmediği çarpıcı bir şekilde

görülmüştür. **Single Linkage** yöntemi, matematiksel olarak en yüksek skoru (**0.64**) üretmesine rağmen; tüm müşterileri tek bir kümeye yığıp aykırı değerleri dışlayarak iş mantığına aykırı bir sonuç vermiştir. Bu durum, kümeleme analizlerinde sadece skorlara güvenilmemesi gerektiğini; kümelerin eleman dağılımının ve ticari anlamlılığının mutlaka insan gözüyle doğrulanması gerektiğini kanıtlamıştır.

- **Hız ve Ölçeklenebilirlik:** Büyük veri setleri için değerlendirildiğinde, algoritmaların işlem maliyetleri arasında uçurum olduğu gözlemlenmiştir. K-Means algoritması, O(n) karmaşıklığı ile 0.11 saniyede sonuç verirken; Hiyerarşik Kümeleme O(n^2) karmaşıklığı nedeniyle 2.42 saniye sürmüştür. Veri seti 10 katına çıktığında Hiyerarşik yöntemin süresi 100 kat artacağı için, bu yöntemin gerçek zamanlı sistemlerde veya milyonluk müşteri verilerinde kullanılması mühendislik açısından sürdürülebilir değildir.
- **Veri Ön İşlemenin Kritik Rolü:** Model başarısının algoritma seçiminden çok, veriye uygulanan Logaritmik Dönüşüm ve Standardizasyon adımlarına bağlı olduğu anlaşılmıştır. Eğer log dönüşümü yapılmasaydı, mesafe temelli çalışan K-Means ve Ward algoritmaları, çok harcama yapan müşterileri aykırı değer sanıp merkezleri kaydıracak ve dengesiz kümeler oluşturacaktı. Başarıyı getiren asıl faktör, verinin dağılımını normale yaklaştıran bu ön işleme mühendisliğidir.
- **Algoritma Davranışı:** BIRCH algoritması, bellek verimliliği odaklı tasarlandığı için küçük veri setinde K-Means kadar keskin ayrımlar yapamamış olsa da veri seti büyüdükçe performansını koruma potansiyeline sahiptir.

Modellerin performans metriklerine göre teknik sıralaması şöyledir:

K-Means (K=2) ≈ Hiyerarşik (Ward) >BIRCH >Hiyerarşik (Average) >Hiyerarşik (Single)

NOT: Son iki model her ne kadar yüksek performanslı gözükse de doğru bir ayırma işlemi yapamamıştır.

Modellerin günlük hayat problemleri ve mühendislik mantığı açısından sıralaması ise şöyledir:

- **K-Means (K=2)**

Neden: GPU/CPU üzerinde **0.11 saniye** gibi inanılmaz bir hızda çalışır, yorumlanması en kolay merkezleri üretir ve yeni gelen bir veriyi anında sınıflandırabilir. Bir e-ticaret sitesinin canlı sisteminde çalışacak müşteri segmentasyonu modülü için endüstri standardı ve tereddütsüz tercih sebebidir.

- **BIRCH**

Neden: Eğer veri seti RAM'e sığmayacak kadar büyükse K-Means belleği zorlayabilir. Bu noktada veriyi özetleyerek ağaç yapısında tutan BIRCH, en güçlü Big Data alternatifidir. Küçük veride K-Means'in gerisinde kalsa da ölçeklenebilirlik açısından mühendislik değeri yüksektir.

- **Hiyerarşik Kümeleme (Ward)**

Neden: K-Means ile benzer kalitede kümeler oluşturmasına rağmen, işlem maliyeti çok yüksektir. Sadece veri setinin küçük olduğu ve hiyerarşik yapının görsel olarak sunulması gereken "butik" analizlerde veya offline raporlamalarda tercih edilebilir. Üretim ortamı için maliyetli bir çözümdür.

- **Hiyerarşik Kümeleme (Single/Average)**

Neden: Gürültüye karşı aşırı hassas olduğu için "Zincirleme Etkisi" yaratır ve dengesiz kümeler oluşturur. Matematiksel skoru ne kadar yüksek olursa olsun, iş problemlerini çözmekte yetersiz kaldığı için son sıradadır.

6. Bu Ödevde Öğrenilenler

- Gözetimli öğrenmeden farklı olarak, bu projede algoritmadan ziyade veriyi hazırlama sürecinin sonucu belirlediği görülmüştür. Milyonlarca satırlık ham işlem verisinin, RFM gibi anlamlı iş metriklerine dönüştürülmeden hiçbir kümeleme algoritmasıyla anlamlı sonuç vermeyeceği deneyimlenmiştir.
- Regresyon projelerinde genellikle silinen aykırı değerlerin, müşteri segmentasyonunda aslında şirketin "en değerli müşterileri" olabileceği fark edilmiştir. Bu nedenle, bu verileri silmek yerine Logaritmik Dönüşümü uygulayarak veriyi algoritmanın anlayacağı formata getirmenin hayati önem taşıdığı öğrenilmiştir.
- Yüksek bir metrik skorunun her zaman "doğru model" anlamına gelmediği kavranmıştır. Matematiksel olarak en yüksek Silhouette skorunu veren Single Linkage yönteminin, veriyi 4337'ye 1 şeklinde ayırarak tamamen işlevsiz bir model ürettiği görülmüş; "Matematiksel Doğruluk" ile "İş Mantığı Uyumu" arasındaki fark net bir şekilde anlaşılmıştır.
- "Test Seti" veya "Gerçek Etiketler" olmadığı için, model başarısının sadece kod çıktılarıyla değil, kümelerin ortalamalarına bakarak yapılan Profil Analizi ile doğrulanması gerektiği öğrenilmiştir. Bir modelin başarısı, oluşturduğu grupların insan gözüyle yorumlanabilir olup olmadığına bağlıdır.
- Teorik olarak bilinen O(n) ve O(n^2) karmaşıklık farklarının pratikteki karşılığı gözlemlenmiştir. K-Means algoritması anlık sonuç verirken, Hiyerarşik Kümeleme'nin, veri arttıkça katlanarak yavaşladığı ve büyük veri projeleri için K-Means veya BIRCH gibi ölçeklenebilir algoritmaların tercih edilmesi gerektiği deneyimlenmiştir.

7. Akademik Dürüstlük ve Yapay Zekâ Kullanımı

Tüm kümeleme modeli ödevi süresince kullanılan tek yapay zekâ asistanı, **Gemini Pro**'dur. Yardım alınan tüm işlemler, aşağıda listelenmiştir:

- **Öznitelik Çıkarımı:** Ham işlem (transaksiyon) verilerinin müşteri bazlı bir yapıya dönüştürülmesi gerekiyordu. groupby ve agg fonksiyonlarını kullanarak Recency, Frequency ve Monetary (RFM) metriklerinin türetilmesi ve veri setinin yeniden yapılandırılması mantığı konusunda kodlama desteği aldım.

- **Aykırı Değer Yönetimi ve Log Dönüşümü:** Müşteri harcama verilerindeki aşırı uç değerlerin silinmesi yerine, veri kaybını önlemek adına nasıl normalize edilebileceğini araştırdım. Yapay zekâ desteğiyle, bu değerleri silmek yerine Logaritmik Dönüşüm uygulamanın ve sonrasında Standartlaştırma yapmanın K-Means algoritması için en doğru yaklaşım olduğunu öğrendim ve koda entegre ettim.

- **Model Doğrulama:** Gözetimsiz öğrenmede GridSearchCV gibi hazır bir yapı olmadığı için, optimum küme sayısını (K) bulmak adına döngüsel bir yapı kurulması gerekiyordu.

Dirsek Yöntemi grafiğinin çizdirilmesi ve Silhouette Skoru'nun yorumlanması (negatif değerlerin veya 1'e yakın değerlerin ne anlama geldiği) konusunda teorik destek aldım.

- **Algoritma Seçimi ve Parametreler:** Özellikle Hiyerarşik Kümeleme algoritmasındaki "Linkage" türlerinin veriyi nasıl böldüğü ve BIRCH algoritmasının parametrelerinin belleği nasıl optimize ettiği konularında teknik bilgi desteği aldım. Single Linkage yönteminin neden zincirleme hatasına yol açtığını bu sayede analiz ettim.
- **Görselleştirme ve Boyut İndirgeme (PCA):** Çok boyutlu (RFM+Diğerleri) verinin 2 boyutta görselleştirilebilmesi için PCA yönteminin kullanımı ve kümelerin saçılım grafiği üzerinde renkli olarak gösterilmesi konusunda kod desteği aldım.
- **Hata Ayıklama ve Kod İyileştirme:** Yellowbrick görselleştirme kütüphanesinin kullanımı ve veri tiplerinin ayarlanması sırasında alınan syntax hatalarının giderilmesi için yapay zekâdan faydalandım.
- **Akademik Dil ve Düzenleme:** Raporun yazım aşamasında, oluşturduğum taslak metinlerin akademik dile uygunluğunun kontrol edilmesi ve anlatım bozukluklarının giderilmesi amacıyla düzenleme desteği alınmıştır. Raporun içeriği, yorumlar ve tüm analizler **tamamen şahsıma aittir.**

Kaynakça

[1] KNN Kütüphane Dokümantasyonu: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[2] KNN ve Random Forest Optimizasyonu için cuML Kütüphane Dokümantasyonu: https://docs.rapids.ai/api/cuml/stable/api

[3] Random Forest Classifier Kütüphane Dokümantasyonu: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[4] XGBoost Kütüphane Dokümantasyonu: https://xgboost.readthedocs.io/en/stable/parameter.html

[5] RFECV Kütüphane Dokümantasyonu: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html

[6] Lineer Regresyon Kütüphane Dokümantasyonu: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[7] Random Forest Regressor Kütüphane Dokümantasyonu: https://scikit--learn-org.translate.goog/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html?_x_tr_sl=en&_x_tr_tl=tr&_x_tr_hl=tr&_x_tr_pto=tc

[8] K-Means Kütüphane Dokümantasyonu: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[9] Hiyerarşik Kümeleme Kütüphane Dokümantasyonu: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

[10]BIRCH Kütüphane Dokümantasyonu: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.Birch.html