SAKARYA
ÜNİVERSİTESİ

Bilgisayar ve Bilişim Bilimleri Fakültesi

Bilgisayar Mühendisliği Bölümü

Makine Öğrenmesi Dersi

Proje Raporu

Mobil Cihazların Fiyat Skalasını Tahmin Eden Makine Öğrenmesi Modeli

GRUP ÜYELERİ

G201210089 Furkan Tataroğlu

G211210079 Umut Bide

G211210047 Mert Eser Meral

G211210007 Ayşenur Yıldırım

Veri Seti Seçimi Ve Açıklaması

Veri Seti: https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification

Bu çalışma kapsamında, mobil cihazların çeşitli teknik özellikleri ile fiyat aralıklarını ilişkilendiren bir veri seti kullanılmıştır. Veri seti, telefonların donanımsal özelliklerini ve kullanıcı deneyimini etkileyen parametreleri içermektedir. Bu parametreler, aşağıdaki gibi detaylandırılmıştır:

- **Batarya Kapasitesi (battery_power)**
  Mobil cihazların batarya kapasitesini mAh cinsinden ifade eden bir değişkendir. Uzun süreli kullanım için kritik bir rol oynayan bu özellik, özellikle kullanıcıların cihaz tercihlerini etkileyen başlıca faktörlerden biridir.
- **Bluetooth Özelliği (blue)**
  Telefonun Bluetooth bağlantısı özelliğine sahip olup olmadığını gösterir. İkili bir değişken olup, 1 mevcut olduğunu, 0 ise olmadığını ifade eder.
- **İşlemci Hızı (clock_speed)**
  Telefonun işlemci hızını GHz cinsinden ölçer. Bu değer, telefonun performansını doğrudan etkileyen önemli bir faktördür.
- **Çift SIM Desteği (dual_sim)**
  Telefonun çift SIM kart desteğine sahip olup olmadığını belirtir. 1 değeri bu desteğin mevcut olduğunu, 0 ise olmadığını ifade eder.
- **Ön Kamera Çözünürlüğü (fc)**
  Telefonun ön kamerasının çözünürlüğünü megapiksel cinsinden ifade eder. Bu değişken, özellikle fotoğraf çekmeyi seven kullanıcılar için önemli bir parametredir.
- **4G Desteği (four_g)**
  Telefonun 4G bağlantı desteği olup olmadığını belirtir. 1 mevcut olduğunu, 0 ise olmadığını ifade eder.
- **Dahili Hafıza (int_memory)**
  Telefonun dahili depolama alanını GB cinsinden ifade eden bir değişkendir. Yüksek hafıza kapasitesi, kullanıcıların daha fazla uygulama ve veri saklamasına olanak tanır.
- **Cihaz Kalınlığı (m_dep)**
  Telefonun kalınlığını santimetre cinsinden ölçen bir değişkendir. Tasarım açısından önemli bir kriterdir.
- **Cihaz Ağırlığı (mobile_wt)**
  Telefonun gram cinsinden ağırlığını ifade eder. Hafif cihazlar, taşınabilirlik açısından kullanıcılar tarafından daha fazla tercih edilir.
- **İşlemci Çekirdek Sayısı (n_cores)**
  Telefonun işlemcisinde bulunan çekirdek sayısını belirtir. Daha fazla çekirdek, cihazın aynı anda birden fazla işlem yapabilme yeteneğini artırır.
- **Arka Kamera Çözünürlüğü (pc)**
  Telefonun arka kamerasının çözünürlüğünü megapiksel cinsinden ifade eder. Fotoğraf kalitesi üzerinde belirleyici bir etkisi vardır.
- **Ekran Çözünürlüğü (px_height ve px_width)**
  Telefon ekranının dikey (px_height) ve yatay (px_width) piksel çözünürlüğünü ifade eder. Ekran kalitesi ve görüntü netliği açısından kritik bir öneme sahiptir.
- **RAM Kapasitesi (ram)**
  Telefonun RAM kapasitesini MB cinsinden ifade eder. RAM, cihazın performansı ve uygulamalar arasındaki geçiş hızını etkileyen en önemli faktörlerden biridir.

- **Ekran Boyutları (sc_h ve sc_w)**
  Telefonun ekran yüksekliği (sc_h) ve genişliği (sc_w) santimetre cinsinden ifade edilir. Kullanıcı deneyimini doğrudan etkileyen bir diğer önemli parametredir.
- **Konuşma Süresi (talk_time)**
  Telefonun tam şarjda sağladığı konuşma süresini saat cinsinden ifade eder. Kullanıcıların şarj dayanıklılığı konusundaki beklentilerini karşılayan bir faktördür.
- **3G Desteği (three_g)**
  Telefonun 3G bağlantı desteği olup olmadığını belirtir. 1 mevcut olduğunu, 0 ise olmadığını ifade eder.
- **Dokunmatik Ekran (touch_screen)**
  Telefonun dokunmatik ekran özelliği olup olmadığını belirtir. 1 mevcut olduğunu, 0 ise olmadığını ifade eder.
- **Wi-Fi Özelliği (wifi)**
  Telefonun Wi-Fi bağlantı özelliği olup olmadığını belirtir. 1 mevcut olduğunu, 0 ise olmadığını ifade eder.
- **Fiyat Aralığı (price_range)**
  Cihazın fiyatını kategorik olarak dört seviyede sınıflandırır:
  - 0: Düşük fiyatlı
  - 1: Orta fiyatlı
  - 2: Yüksek fiyatlı
  - 3: Çok yüksek fiyatlı

Bu veri seti, telefonların farklı donanım özelliklerinin fiyat aralıkları ile nasıl ilişkilendirildiğini anlamak ve modelleme yapmak için kullanılmaktadır. Bu özelliklerin detaylı analizi, müşteri tercihlerini etkileyen temel faktörleri belirlemek açısından son derece önemlidir.

Veri Setinin Yapısı ve Genel Bilgileri

Bu çalışma için kullanılan veri seti, mobil cihazların çeşitli özelliklerini ve fiyat aralıklarını analiz etmeye yönelik bilgiler içermektedir. Aşağıda veri setinin yapısına ilişkin detaylar verilmiştir:

1. Veri Setinin Boyutu

- **Gözlem Sayısı (Satır):** Veri setinde toplam **2000** örnek bulunmaktadır.
- **Özellik Sayısı (Sütun):** Veri seti **21 farklı özellik** içermektedir.

2. Veri Türleri

Veri setindeki sütunların türlerine göre dağılımı şu şekildedir:

- **Sayısal Değişkenler:** Batarya kapasitesi (battery_power), işlemci hızı (clock_speed), RAM (ram) gibi sürekli değer alan değişkenlerdir.

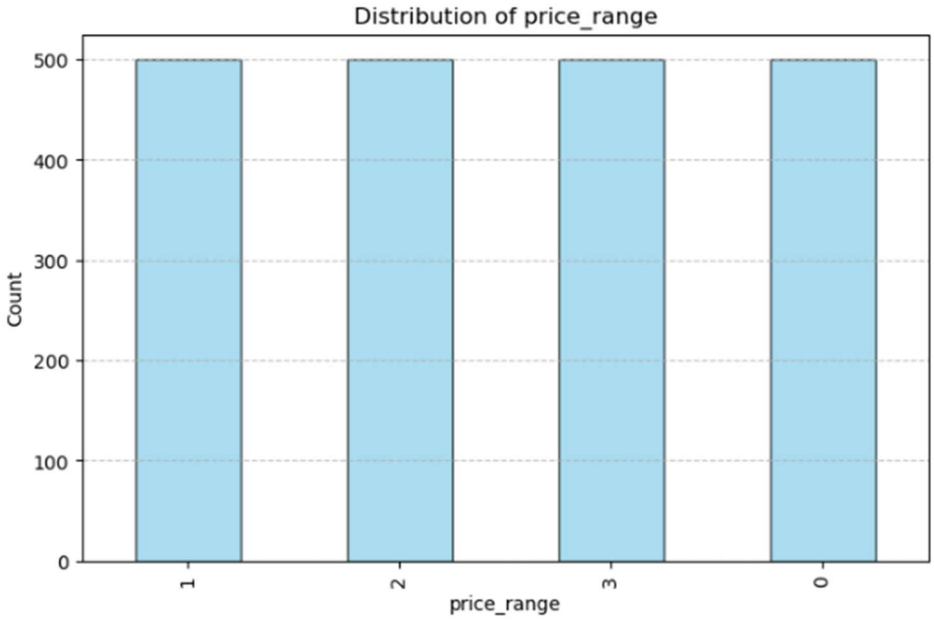
- **Kategorik Değişken:** Hedef değişkenimiz olan **price_range**, fiyat aralıklarını dört kategoride sınıflandırmaktadır (0: Düşük, 1: Orta, 2: Yüksek, 3: Çok Yüksek).

3. Hedef Değişken: price_range

- **Hedef Değişken (price_range):** Bu değişken, telefonların fiyatlarını kategorik olarak sınıflandırmaktadır.
  - 0: Düşük fiyatlı
  - 1: Orta fiyatlı
  - 2: Yüksek fiyatlı
  - 3: Çok yüksek fiyatlı

4. Sınıf Dağılımı

Hedef değişkenin (price_range) sınıf dağılımını analiz ederek, verinin dengeli olup olmadığını kontrol ettik:



5. Eksik Veri ve Temizlik

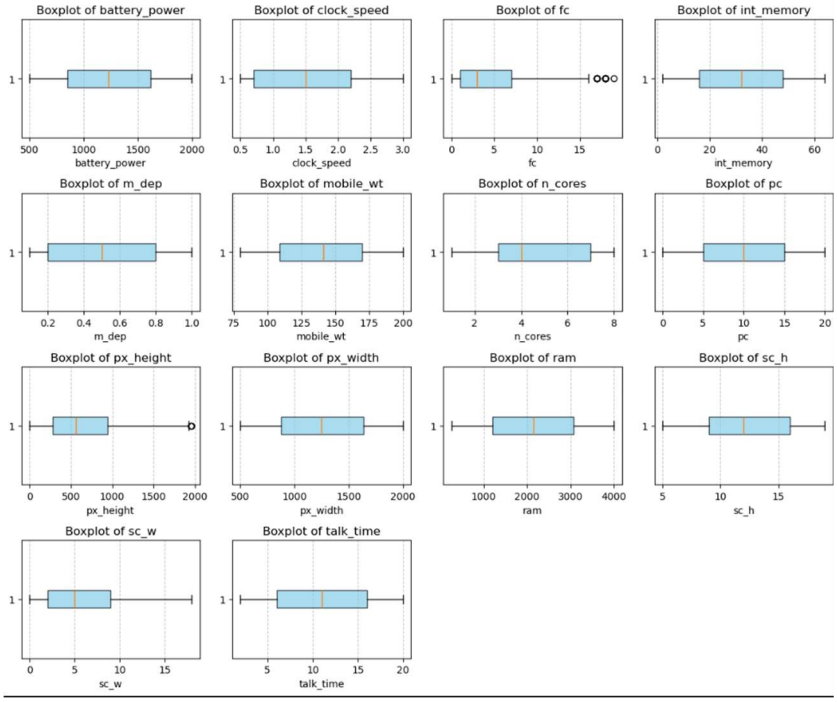
Veri Setinin Temizlenmesi

Öncelikle, veri setimi özenle inceledik ve herhangi bir eksik veri olmadığını gözlemledik.

```
data.isnull().sum()
#BOS DATA YOK
```

battery_power	0
blue	0
clock_speed	0
dual_sim	0
fc	0
four_g	0
int_memory	0
m_dep	0
mobile_wt	0
n_cores	0
pc	0
px_height	0
px_width	0
ram	0
sc_h	0
sc_w	0
talk_time	0
three_g	0
touch_screen	0
wifi	0
price_range	0

Sayısal verilerin histogram grafiklerine bakarak ,verilerin tutarlı ve anlamlı bir dağılıma sahip olduğunu gözlemledik. Bu nedenle, veriyi "Temiz Veri " olarak adlandırarak başlangıç analizlerimizi gerçekleştirdik. Temiz veriyi kullanarak çeşitli görselleştirmeler ve temel analizler yaptık. Özellikle histogramlar, boxplotlar kullanarak her bir özelliğin dağılımını ve etkileşimlerini inceledik. Bu analizler sonucunda, veri setinin düzgün bir şekilde dağıldığını ve herhangi bir uç değerin olmadığını gördük.



PCA ve KPCA Karşılaştırması

Veri setimizde hangi özellik çıkarma algoritmasını kullanmamız gerektiğini bulmak için bu iki algoritmanın çıktılarını KNN algoritmasında test ettik. Doğruluk değerlerimiz sonucu KPCA kullanmaya karar verdik.

```
#KNN PCA
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y, test_size=0.3, random_state=43, stratify=y)


knn_pca = KNeighborsClassifier(n_neighbors=5)  # Default k=5
knn_pca.fit(X_train_pca, y_train)

y_pred_pca = knn_pca.predict(X_test_pca)

accuracy_pca = accuracy_score(y_test, y_pred_pca)
classification_rep_pca = classification_report(y_test, y_pred_pca)

accuracy_pca, classification_rep_pca
```

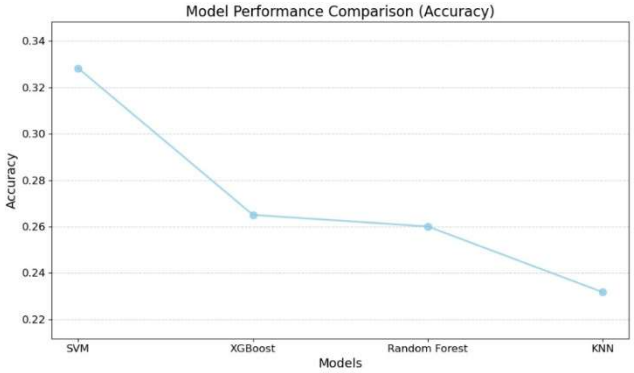
```
(0.79,
 '              precision    recall  f1-score   support\n\n           0       0.84      0.93      0.89       150\n           1       0.73      0.69
0.71       150\n           2       0.71      0.68      0.69       150\n           3       0.87      0.86      0.87       150\n\n    accuracy
0.79       600\n   macro avg       0.79      0.79      0.79       600\nweighted avg       0.79      0.79      0.79       600\n')
```

```
#KNN KPCA
X_train_kpca, X_test_kpca, y_train, y_test = train_test_split(X_kpca, y, test_size=0.3, random_state=43, stratify=y)

knn_kpca = KNeighborsClassifier(n_neighbors=5)  # Default k=5
knn_kpca.fit(X_train_kpca, y_train)

y_pred_kpca = knn_kpca.predict(X_test_kpca)

accuracy_kpca = accuracy_score(y_test, y_pred_kpca)
classification_rep_kpca = classification_report(y_test, y_pred_kpca)

accuracy_kpca, classification_rep_kpca
```

```
(0.9066666666666666,
 '              precision    recall  f1-score   support\n\n           0       0.92      0.96      0.94       150\n           1       0.90      0.87
0.89       150\n           2       0.90      0.85      0.87       150\n           3       0.91      0.94      0.92       150\n\n    accuracy
0.91       600\n   macro avg       0.91      0.91      0.91       600\nweighted avg       0.91      0.91      0.91       600\n')
```

Normalizasyon ve Standardizasyon Testi

Sayısal verilerimize ayrı ayrı normalizasyon ve standardizasyon işlemleri yaptık. İşlemler ve testler sonucu normalizasyon ve standardizasyon yapmanın algoritmalarımızdaki doğruluk değerini kötü etkilediğini gözlemedik.



Veri Setini Bozma

Veri setimizin kendisi oldukça temiz olduğu için elle bozmaya karar verdik. Verimize eksik ve uç değerler ekledik ve. Örneğin, belirli bir özellik için minimum veya maksimum değerlerin dışında rastgele bir değer seçtik. Ayrıca, bazı özelliklerde anlamlı olmayan ilişkiler oluşturarak veri setinin anlamlı bir yapısını bozduk; örneğin, ekran çözünürlüğünü orantısız bir şekilde artırdık veya değiştirdik. Bu süreçte, verinin anlamlı bir yapısını bozarak, gerçek hayatta karşılaşılabilecek hatalı uç değerleri simüle etmeye çalıştık.

Veri Ön İşleme ve Veri Analizi

Öncelikle tekrar eden verileri temizledik.

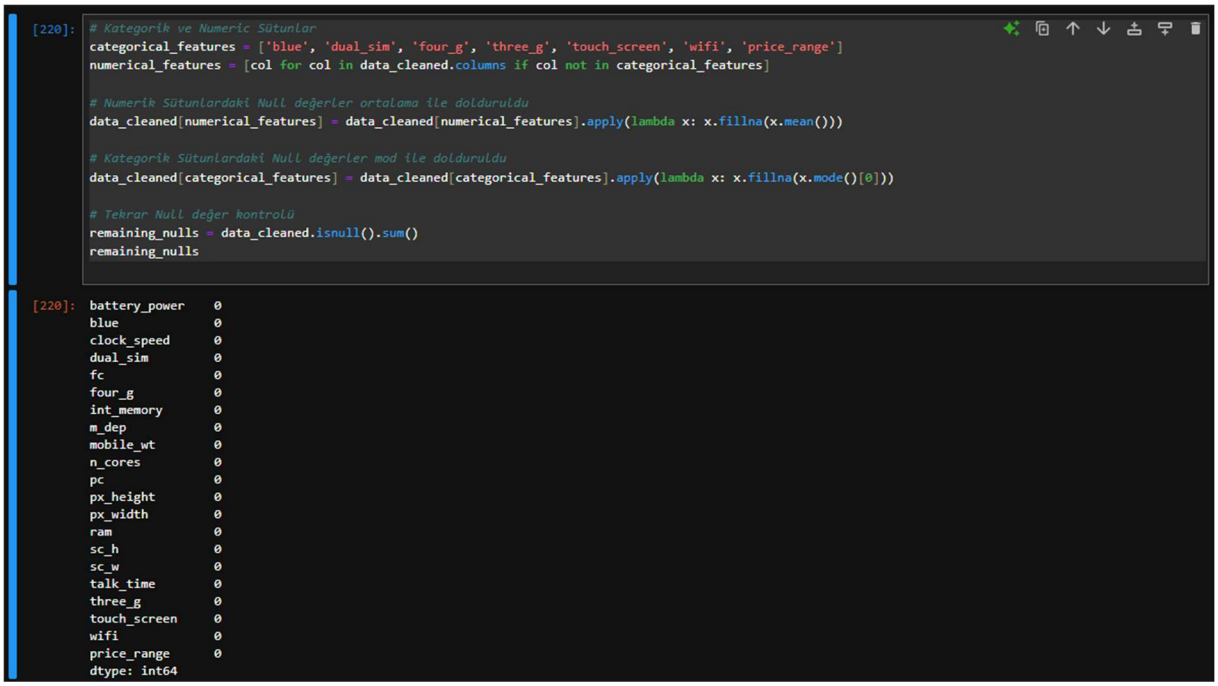
```
# Duplicate veri temizliği
data_cleaned = data_cleaned[data_cleaned.duplicated(keep='first') == False]
data_cleaned
```

Ardından veri setimizde eksik değer kontrolü yaptık.

```
#Null Değer Kontrolü
data_cleaned.isnull().sum()
```

```
battery_power     5
blue             10
clock_speed       7
dual_sim          5
fc                0
four_g            3
int_memory        9
m_dep             5
mobile_wt         6
n_cores           2
pc                4
px_height         5
px_width          1
ram               8
sc_h              3
sc_w              1
talk_time         1
three_g           6
touch_screen      6
wifi              8
price_range      14
dtype: int64
```

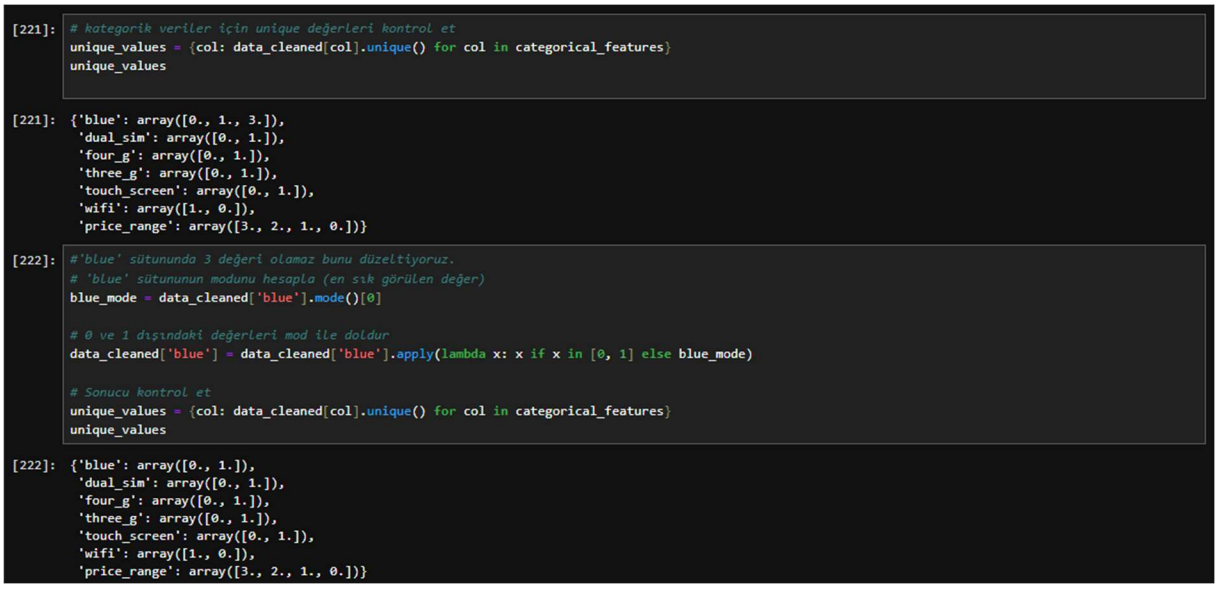
Daha sonra kategorik ve numerik sütunlarımızı ayırdık. Ardından numerik sütunlardaki null değerleri ortalama alarak, kategorik sütunlardaki null değerleri mod alarak doldurduk. Tekrar null değer kontrolü yaptığımızda işlemin başarılı olduğunu tespit ettik.



```
[220]: # Kategorik ve Numeric Sütunlar
categorical_features = ['blue', 'dual_sim', 'four_g', 'three_g', 'touch_screen', 'wifi', 'price_range']
numerical_features = [col for col in data_cleaned.columns if col not in categorical_features]

# Numerik Sütunlardaki Null değerler ortalama ile dolduruldu
data_cleaned[numerical_features] = data_cleaned[numerical_features].apply(lambda x: x.fillna(x.mean()))

# Kategorik Sütunlardaki Null değerler mod ile dolduruldu
data_cleaned[categorical_features] = data_cleaned[categorical_features].apply(lambda x: x.fillna(x.mode()[0]))

# Tekrar Null değer kontrolü
remaining_nulls = data_cleaned.isnull().sum()
remaining_nulls
```

```
[220]: battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

Kategorik veriler için unique değerleri çıktı olarak aldık. ‘blue’ sütununda yalnızca 0 ve 1 değeri olması gerekirken 3 değerlerinin bulunduğunu tespit ettik. Bu değerleri mod alarak düzelttik.



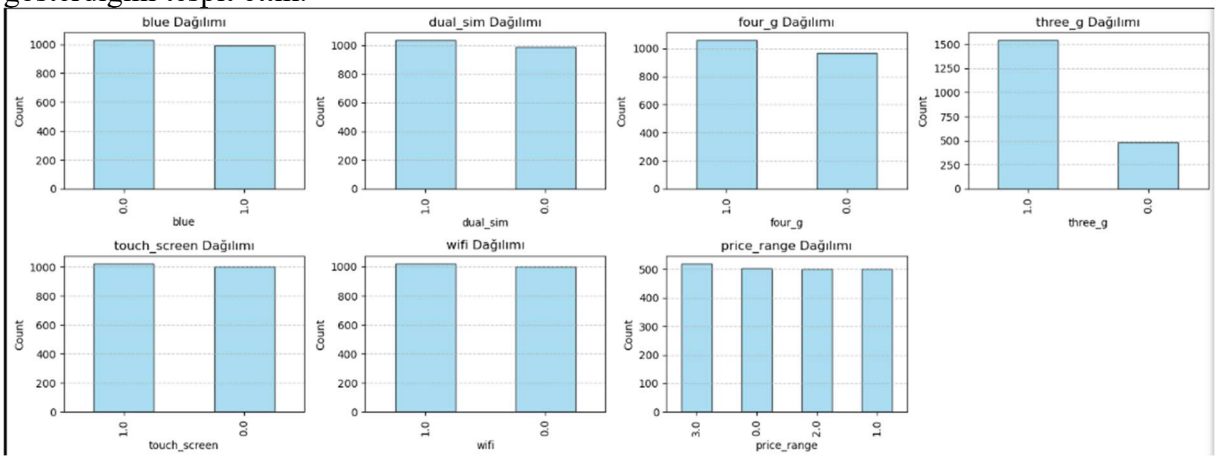
```
[221]: # kategorik veriler için unique değerleri kontrol et
unique_values = {col: data_cleaned[col].unique() for col in categorical_features}
unique_values
```

```
[221]: {'blue': array([0., 1., 3.]),
 'dual_sim': array([0., 1.]),
 'four_g': array([0., 1.]),
 'three_g': array([0., 1.]),
 'touch_screen': array([0., 1.]),
 'wifi': array([1., 0.]),
 'price_range': array([3., 2., 1., 0.])}
```

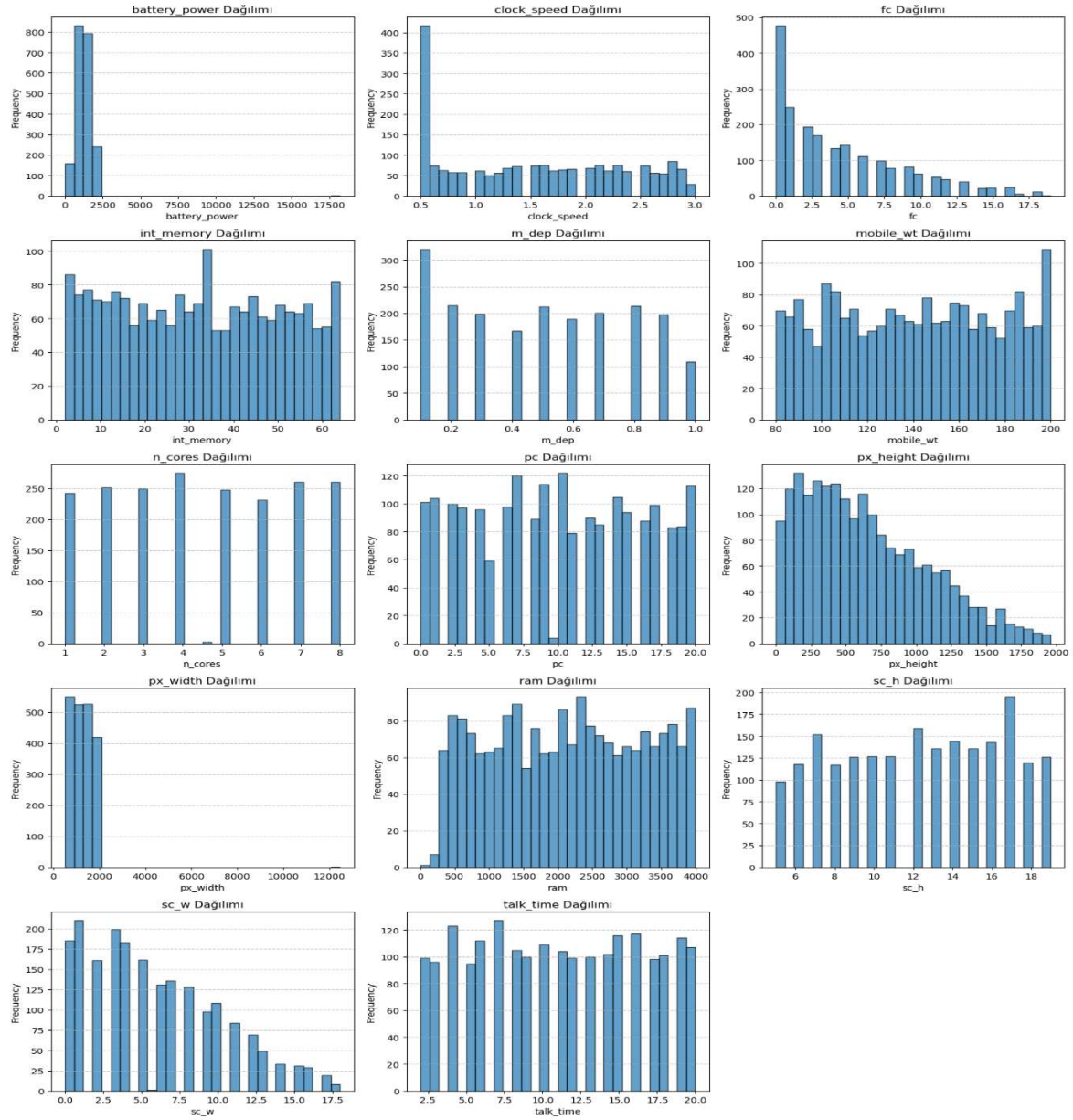
```
[222]: #'blue' sütununda 3 değeri olamaz bunu düzeltiyoruz.
# 'blue' sütununun modunu hesapla (en sık görülen değer)
blue_mode = data_cleaned['blue'].mode()[0]

# 0 ve 1 dışındaki değerleri mod ile doldur
data_cleaned['blue'] = data_cleaned['blue'].apply(lambda x: x if x in [0, 1] else blue_mode)

# Sonucu kontrol et
unique_values = {col: data_cleaned[col].unique() for col in categorical_features}
unique_values
```

```
[222]: {'blue': array([0., 1.]),
 'dual_sim': array([0., 1.]),
 'four_g': array([0., 1.]),
 'three_g': array([0., 1.]),
 'touch_screen': array([0., 1.]),
 'wifi': array([1., 0.]),
 'price_range': array([3., 2., 1., 0.])}
```

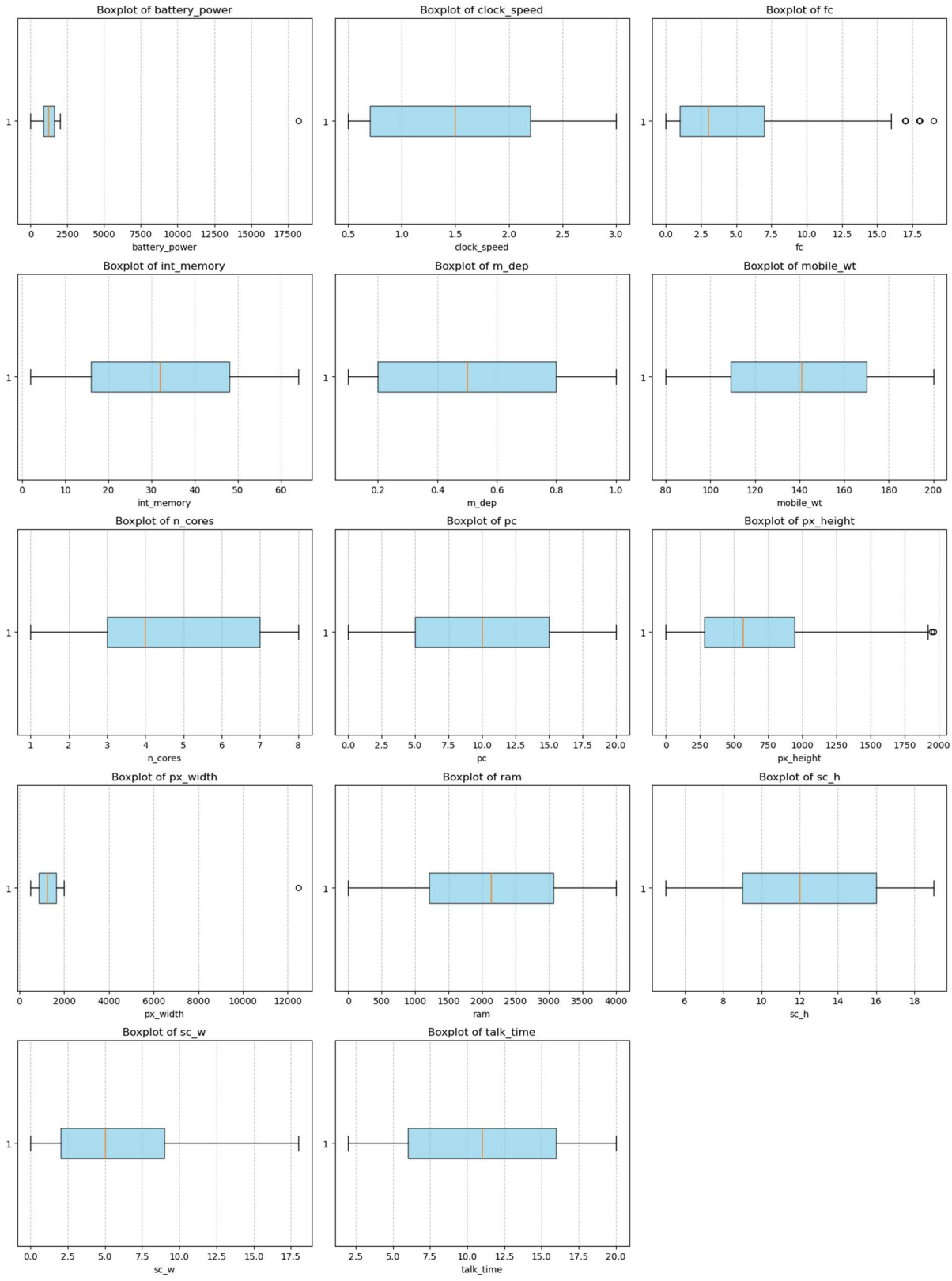
Kategorik verilerimizin dağılımlarını görselleştirdik. Genel olarak verilerin dengeli dağılım gösterdiğini tespit ettik.



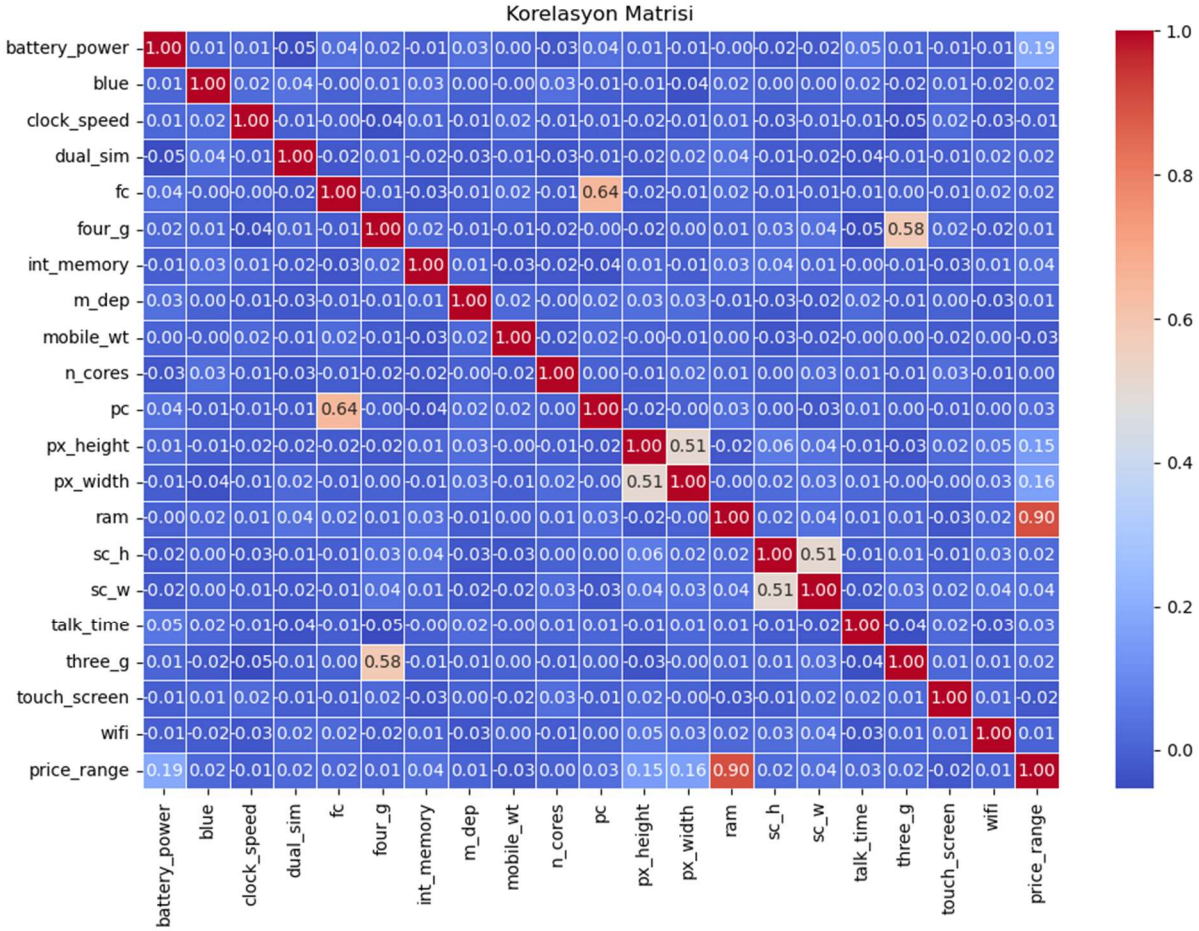
Numerik verilerimizin de dağılımlarını görselleştirdik. Bazı verilerde uç değerler tespit ettik.



Sonrasında numerik veriler için uç değer analizi yaptık ve görselleştirdik. Z-score ile tespit ettiğimiz uç değerleri temizledik.

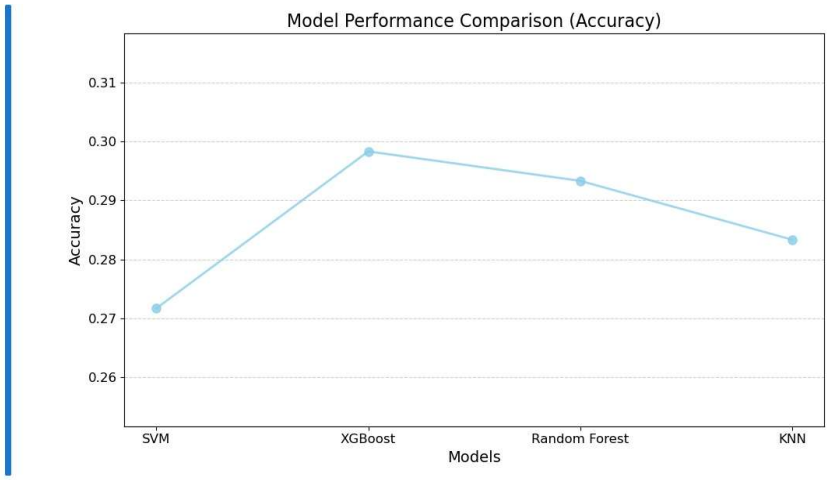


Korelasyon Matrisi ısı haritası oluşturarak numerik özellikler arasındaki ilişkileri gözlemledik. Bu ısı haritası, değişkenler arasındaki pozitif ve negatif ilişkileri görselleştirerek, veri üzerinde daha derinlemesine bir analiz sağlar. Hedef değişken ile "ram" değişkeni arası ilişkinin çok yüksek olduğunu fark ettik. "fc" ve "pc" arasında ve "four_g" ve "three_g" arasındaki ilişki yüksek olduğundan "fc" ile "three_g" özelliklerini veri setimizden kaldırdık.

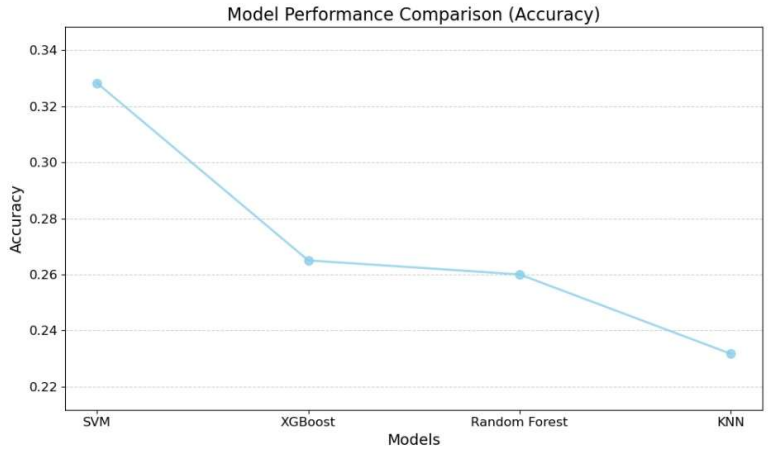


Veri Standardizasyonu ve Normalizasyonu

- **Standartizasyon**: Sayısal değişkenler için **StandardScaler** kullanılarak standartlaştırma işlemi gerçekleştirilmiştir. Standartlaştırma, verilerin her bir değişken için ortalama ve varyansa göre merkezi hale getirilmesini sağlar.



- **Normalizasyon**: Aynı sayısal değişkenler için **MinMaxScaler** kullanılarak normalizasyon işlemi gerçekleştirilmiştir. Normalizasyon, verilerin belirli bir aralıkta (genellikle 0-1) ölçeklendirilmesini sağlar.

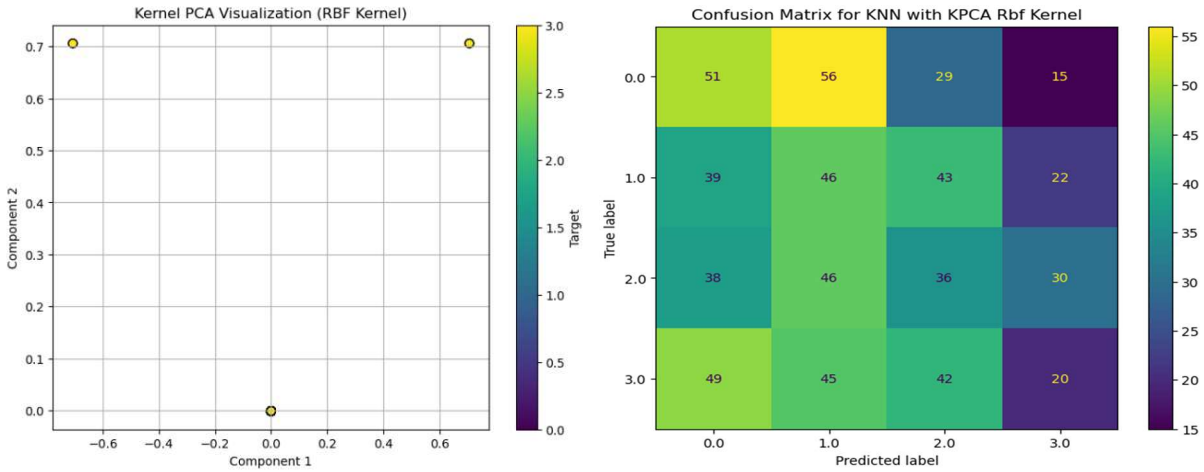


Normalizasyon ve Standardizasyon sonucu model performansları çok düşük çıktığı için bu yöntemleri kullanmadık.

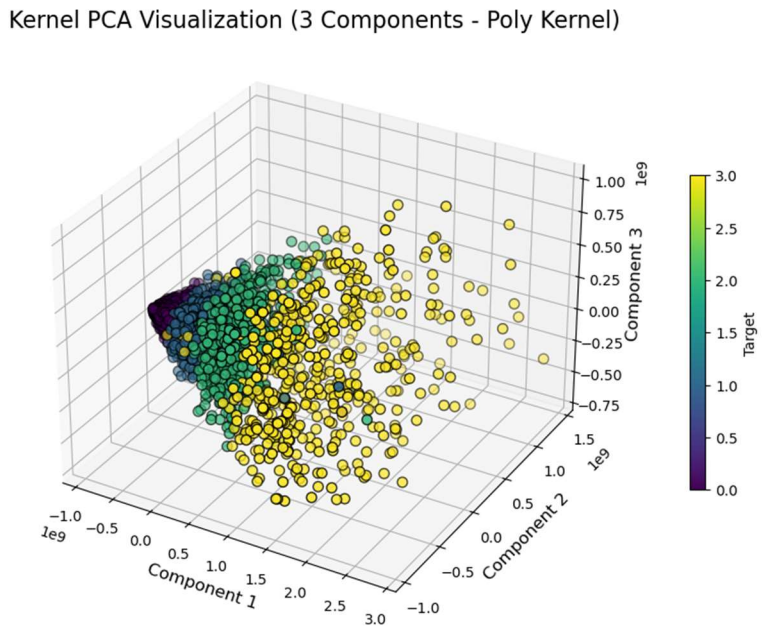
Kernel PCA Uygulaması

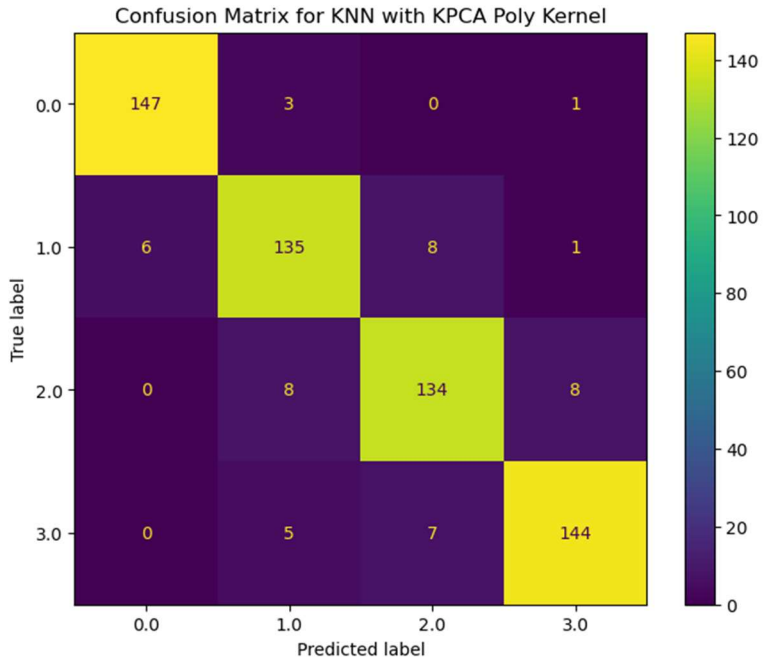
Öncelikle Rbf Kernel ile kpca uygulayıp KNN algoritmasında sonucu değerlendirdik. Sonucun çok düşük çıkması sonucu poly Kernel ile tekrar kpca uyguladık. Bu sefer component sayısını 3 seçip doğruluk oranını yükselttik.

Rbf Kernel Görselleştirmesi ve Karışıklık Matrisi



Poly Kernel Görselleştirmesi ve Karışıklık Matrisi





Model Eğitimi

Model eğitimi için KNN, SVM, XGBoost, Random Forest, Naive Bayes modellerini kullandık.

KNN Modeli

1. Doğruluk Değeri ve Classification Report

Doğruluk Oranı: %92.26

Bu oran, KNN modelinin KPCA Poly Kernel sonrası verilerde başarılı bir sınıflandırma performansı sergilediğini göstermektedir.

Classification Report:

Precision: Tüm sınıflar arasında dengeli (%89-%96).

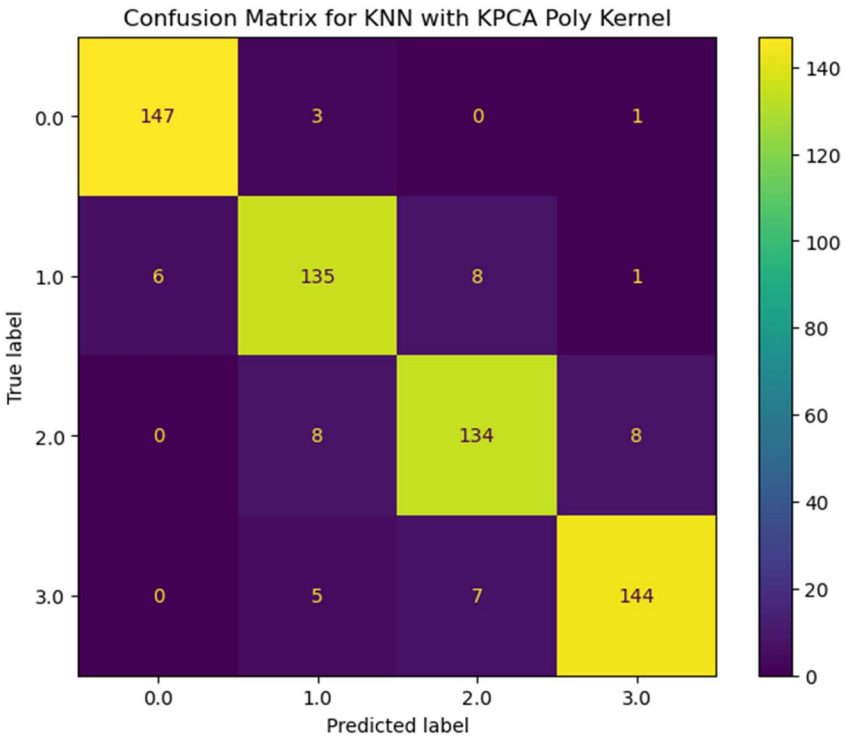
En yüksek precision, sınıf 0'dadır (%96).

Recall: Tüm sınıflar için dengelidir ve %89 ile %97 arasında değişmektedir.

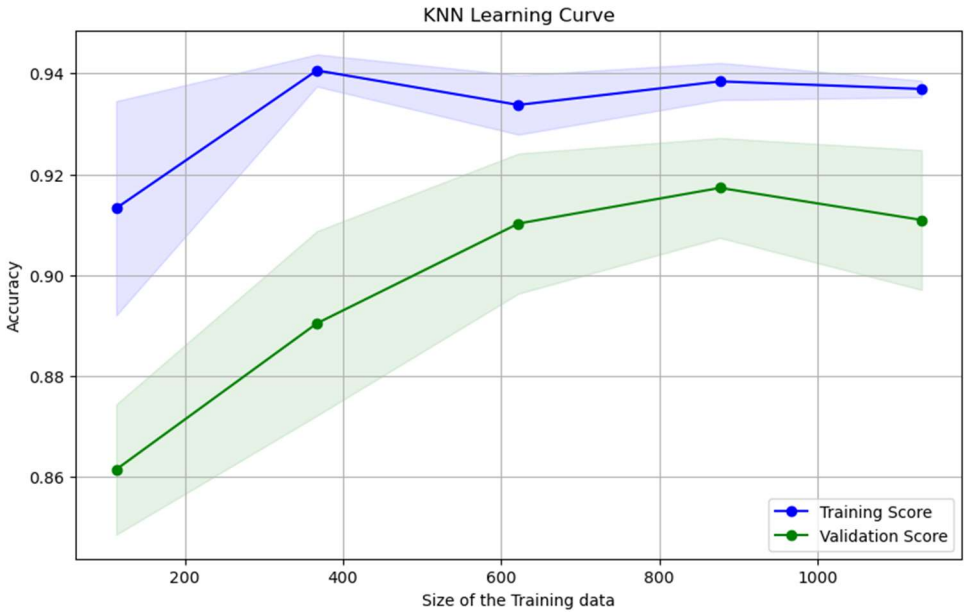
En yüksek recall sınıf 0'dadır (%97).

F1-Score: Ortalama %92, bu da modelin genel sınıflandırma performansının güçlü ve dengeli olduğunu gösterir.

2. Confusion Matrix (Karışıklık Matrisi)



3. Learning Curve (Öğrenme Eğrisi)



Eğitim Skoru (Mavi Eğri): Yüksek ve sabit (%94-%93). Model eğitim verisine iyi uyum sağlıyor.

Doğrulama Skoru (Yeşil Eğri): Eğitim veri boyutu arttıkça düzenli olarak yükseliyor ve %92'ye ulaşıyor.

Eğriler Arasındaki Fark: Başlangıçta büyükken, eğitim boyutu arttıkça fark kapanıyor. Model overfitting yapmıyor ve genelleme yeteneği artıyor.

Genel Sonuçlar

- **Performans:** KNN modeli güçlü bir sınıflandırma başarısı göstermiştir.
- **Hata Analizi:** Yanlış tahminler azdır, model genelde dengeli bir performans sergilemektedir.
- **Genelleme Yeteneği:** Model overfitting yapmamakta, daha fazla veriyle genelleme kabiliyeti artmaktadır.

Bu analizler, KNN modelinin güçlü bir performansa sahip olduğunu ve güvenle kullanılabileceğini göstermektedir.

Random Forest Modeli

1. Doğruluk Değeri ve Classification Report

Doğruluk Oranı: %88.47

Random Forest modeli, KPCA dönüşümünden sonra %88.47 doğruluk oranına ulaşmıştır. Bu oran, modelin test veri setindeki genel performansını temsil eder.

Classification Report:

Precision: Sınıf 0 için en yüksek (%94), sınıf 2 için en düşük (%83).

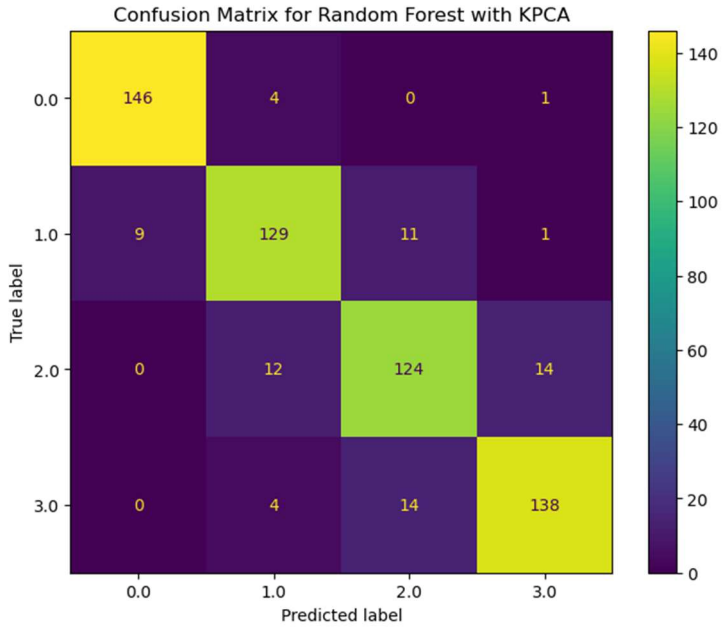
Model, özellikle sınıf 0'daki tahminlerinde daha başarılıdır.

Recall: Sınıf 0 için %97, diğer sınıflar için %83-%90 arasında.

Model, sınıf 0 örneklerini ayırt etmede diğer sınıflara göre daha başarılıdır.

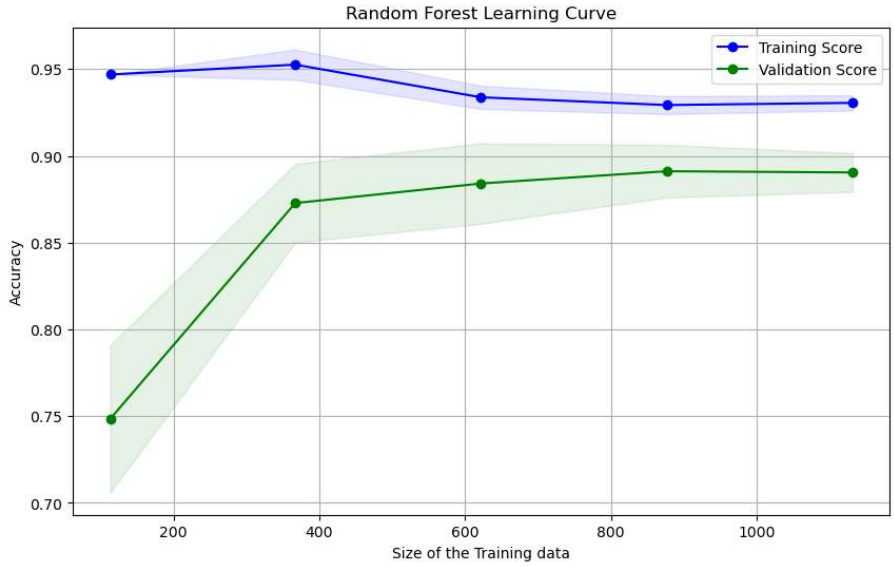
F1-Score: Tüm sınıflar için ortalama %88, modelin genel performansının dengeli olduğunu gösteriyor.

2. Confusion Matrix (Karışıklık Matrisi)



Model, genelde sınıfları doğru tahmin etse de sınıf 2 ve sınıf 3 arasında bazı karışıklıklar mevcuttur. Bu durum, modelin bu sınıfları ayırmakta biraz zorlandığını göstermektedir.

3. Learning Curve (Öğrenme Eğrisi)



Eğitim Skoru (Mavi Eğri):

Eğitim seti boyutu arttıkça skor hafifçe düşmekte ve %95 civarında sabitlenmektedir. Bu, modelin eğitim verisine iyi uyum sağladığını, ancak overfitting'e karşı dayanıklı olduğunu gösterir.

Doğrulama Skoru (Yeşil Eğri):

Başlangıçta (%70 civarı) düşük bir seviyede başlar, ancak eğitim veri boyutu arttıkça %88'e yükselir ve dengelenir. Bu, modelin daha fazla veri ile genelleme yeteneğini artırdığını gösterir.

Eğriler Arasındaki Fark:

Eğitim ve doğrulama skorları arasındaki fark azalmaktadır, bu da modelin overfitting yapmadığını ve dengeli bir genelleme performansı sergilediğini gösterir.

Genel Sonuçlar

- **Performans:** Random Forest modeli, %88.47 doğruluk oranı ile başarılı bir performans sergilemektedir.
- **Hata Analizi:** Sınıf 2 ve sınıf 3 arasındaki karışıklık modelin iyileştirilebilecek bir yönüdür.
- **Genelleme Yeteneği:** Model, overfitting yapmamakta ve daha fazla veriyle doğrulama performansını artırmaktadır.

Bu sonuçlar, Random Forest modelinin genelde dengeli ve güçlü bir sınıflandırma performansı sergilediğini göstermektedir.

XGBoost Modeli

1. Doğruluk Değeri ve Classification Report

Doğruluk Oranı: %91.59

XGBoost modeli, KPCA sonrası veri üzerinde oldukça yüksek bir doğruluk oranına ulaşmıştır.

Classification Report:

Precision: Tüm sınıflar için yüksek ve dengeli (%89-%97 arasında).

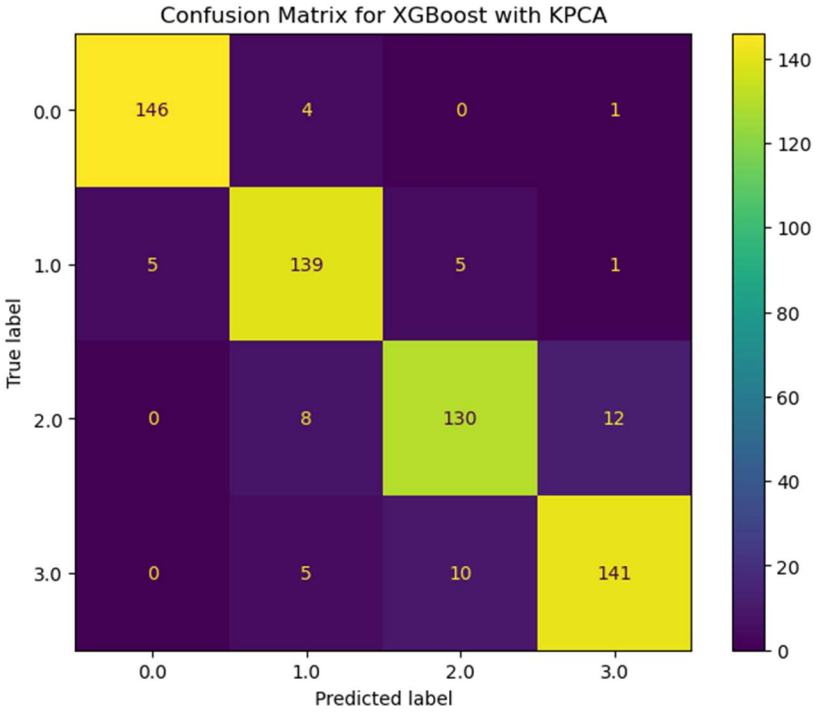
En yüksek precision, sınıf 0 (%97).

Recall: Sınıf 0 ve sınıf 1 için oldukça yüksek (%97 ve %93).

Sınıf 2 (%87) ve sınıf 3 (%90) için hafif bir düşüş var.

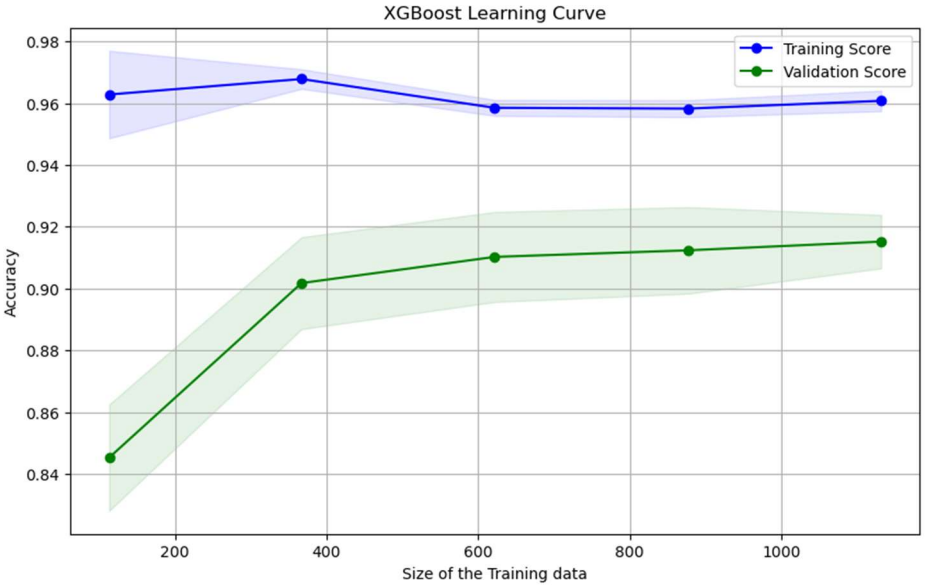
F1-Score: Ortalama %92 ile genel performansın iyi olduğunu göstermektedir.

2. Confusion Matrix (Karışıklık Matrisi)



Yanlış tahminler azdır ve model genel olarak sınıfları iyi ayırmaktadır. Ancak sınıf 2 ve 3 arasında bir miktar karışıklık görülmektedir.

3. Learning Curve (Öğrenme Eğrisi)



Eğitim Skoru (Mavi Eğri):

Başlangıçta çok yüksek (%97-%98) ve veri arttıkça %96 civarında sabitlenmektedir.

Doğrulama Skoru (Yeşil Eğri):

Küçük veri boyutlarında düşük (%84), ancak veri boyutu arttıkça %92'ye kadar yükselmiştir.

Eğriler Arasındaki Fark:

Eğitim ve doğrulama skorları arasındaki fark, eğitim veri boyutunun artmasıyla azalmaktadır.

Bu durum, modelin overfitting yapmadığını ve genelleme kapasitesinin güçlü olduğunu gösterir.

Genel Sonuçlar

- **Performans:** XGBoost modeli, %91.59 doğruluk oranı ve güçlü precision/recall değerleriyle oldukça başarılıdır.
- **Hata Analizi:** Sınıf 2 ve 3 arasında karışıklık az da olsa dikkat çekicidir. Ancak genel olarak sınıflar iyi ayrılmıştır.
- **Genelleme Yeteneği:** Model overfitting yapmamaktadır ve daha fazla veri ile doğrulama performansı artmıştır.

Bu analizler, XGBoost modelinin sınıflandırma görevlerinde etkili bir performans gösterdiğini açıkça ortaya koymaktadır.

SVM Modeli

1. Doğruluk Değeri ve Classification Report

Doğruluk Oranı: %92.92

SVM modeli, KPCA sonrası verilerde yüksek bir doğruluk oranına sahiptir.

Classification Report:

Precision: Sınıflar için dengeli (%91-%95 arasında).

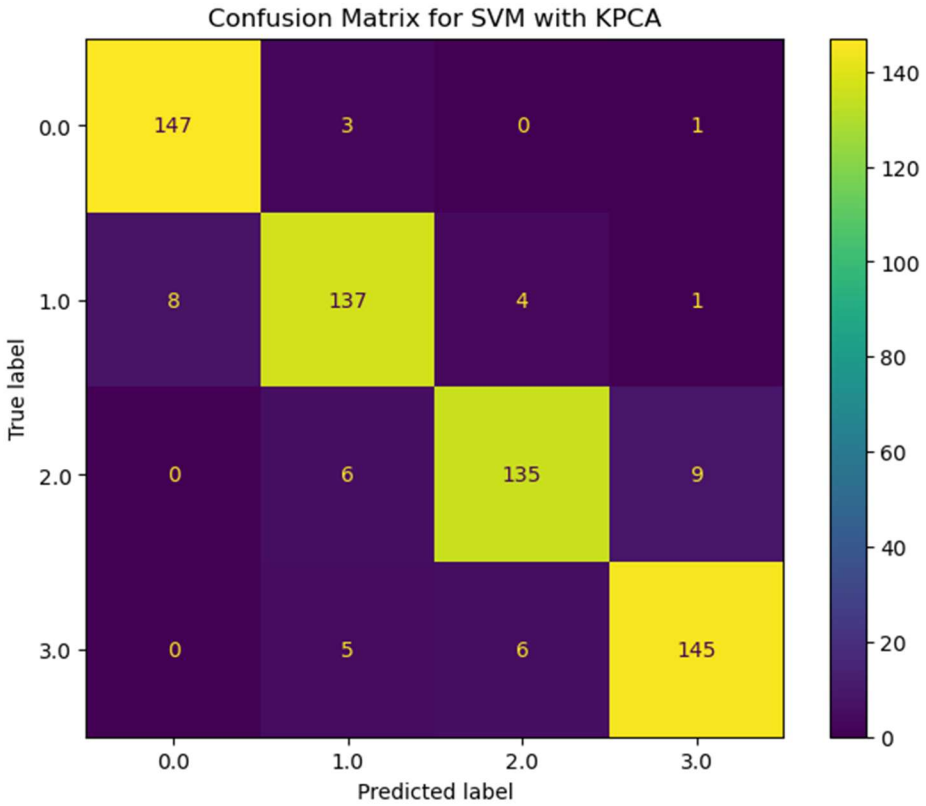
En yüksek precision sınıf 0'da (%95), en düşük sınıf 1'de (%91).

Recall: Sınıflar arasında %90-%97 arasında değişmektedir.

Sınıf 0 için en yüksek (%97), diğer sınıflar da dengelidir.

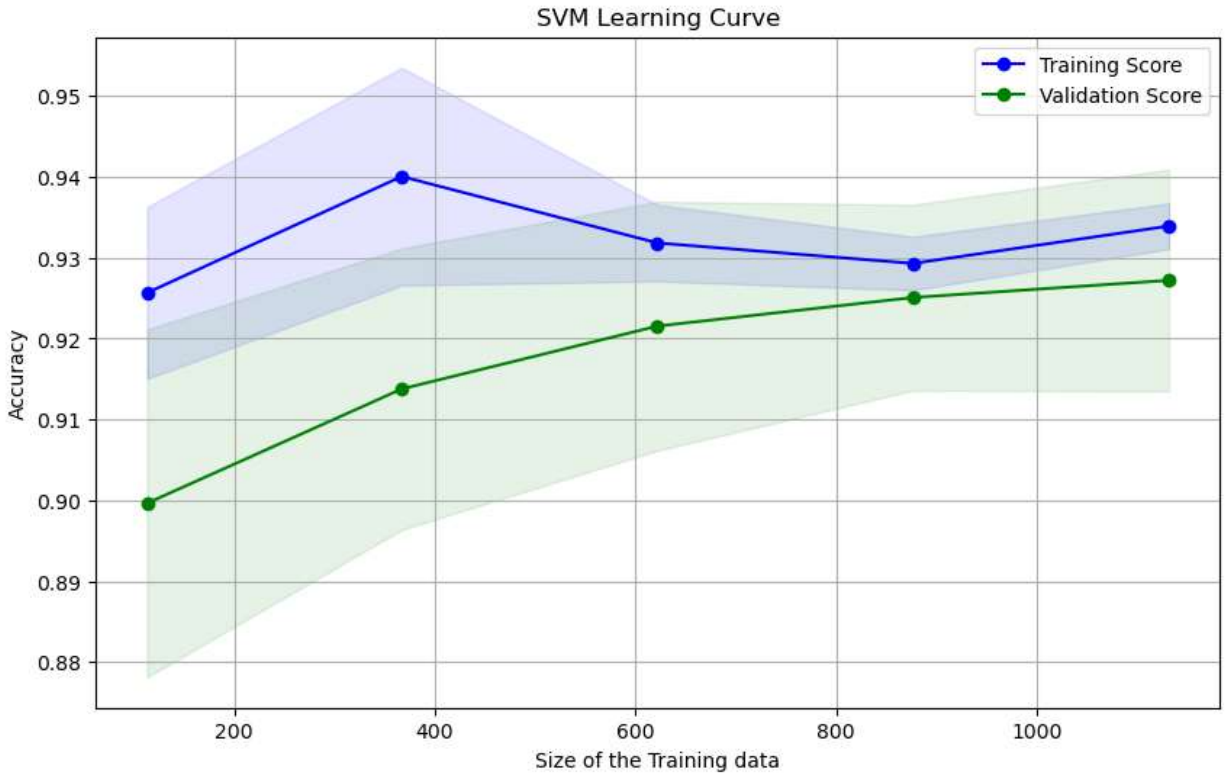
F1-Score: Ortalama %93, bu da modelin genel performansının güçlü olduğunu göstermektedir.

2. Confusion Matrix (Karışıklık Matrisi)



Genel olarak, yanlış tahmin sayısı azdır ve model doğru sınıflandırma yapma konusunda başarılıdır.

3. Learning Curve (Öğrenme Eğrisi)



Eğitim Skoru (Mavi Eğri):

Başlangıçta yüksek (%94 civarı) ve eğitim veri boyutu arttıkça sabit kalmaktadır. Bu, modelin eğitim verisine iyi uyum sağladığını göstermektedir.

Doğrulama Skoru (Yeşil Eğri):

Küçük veri boyutlarında düşük (%90), ancak eğitim veri boyutu arttıkça doğrulama skoru yükselmiş ve %93'e ulaşmıştır.

Eğriler Arasındaki Fark:

Eğitim ve doğrulama skorları arasındaki fark küçüktür ve eğitim veri boyutunun artmasıyla daha da azalmaktadır.

Bu durum, modelin overfitting yapmadığını ve genelleme performansının güçlü olduğunu gösterir.

Genel Sonuçlar

- **Performans:** SVM modeli, %92.92 doğruluk oranı ve dengeli sınıflandırma performansıyla oldukça başarılıdır.
- **Hata Analizi:** Model, sınıflar arasında küçük yanlış tahminler yapmaktadır. Örneğin, sınıf 1 ve 0 arasında az miktarda karışıklık vardır.
- **Genelleme Yeteneği:** Model overfitting yapmamış ve doğrulama performansı eğitim veri boyutu arttıkça iyileşmiştir.

Bu analizler, SVM modelinin KPCA sonrası verilerde başarılı ve dengeli bir sınıflandırma performansı sergilediğini göstermektedir.

Naive Bayes Modeli

1. Doğruluk Değeri ve Classification Report

Doğruluk Oranı: %88.79

Naive Bayes modeli, KPCA dönüşümünden sonra tatmin edici bir doğruluk oranı sağlamıştır.

Classification Report:

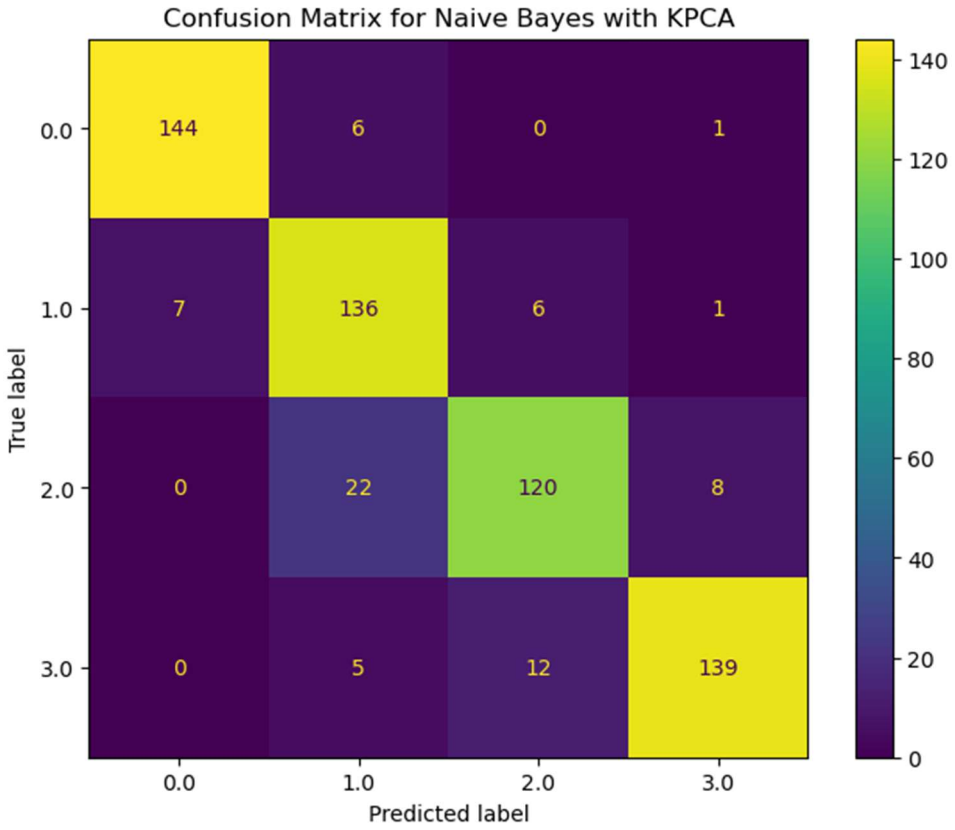
Precision: Sınıflar arasında dengeli bir dağılım (%87-%95).

En yüksek precision sınıf 0'dadır (%95), en düşük ise sınıf 2'dedir (%87).

Recall: Sınıf 0 ve sınıf 1 için oldukça yüksek (%95 ve %91), sınıf 2 (%80) ve sınıf 3 (%89) için daha düşüktür.

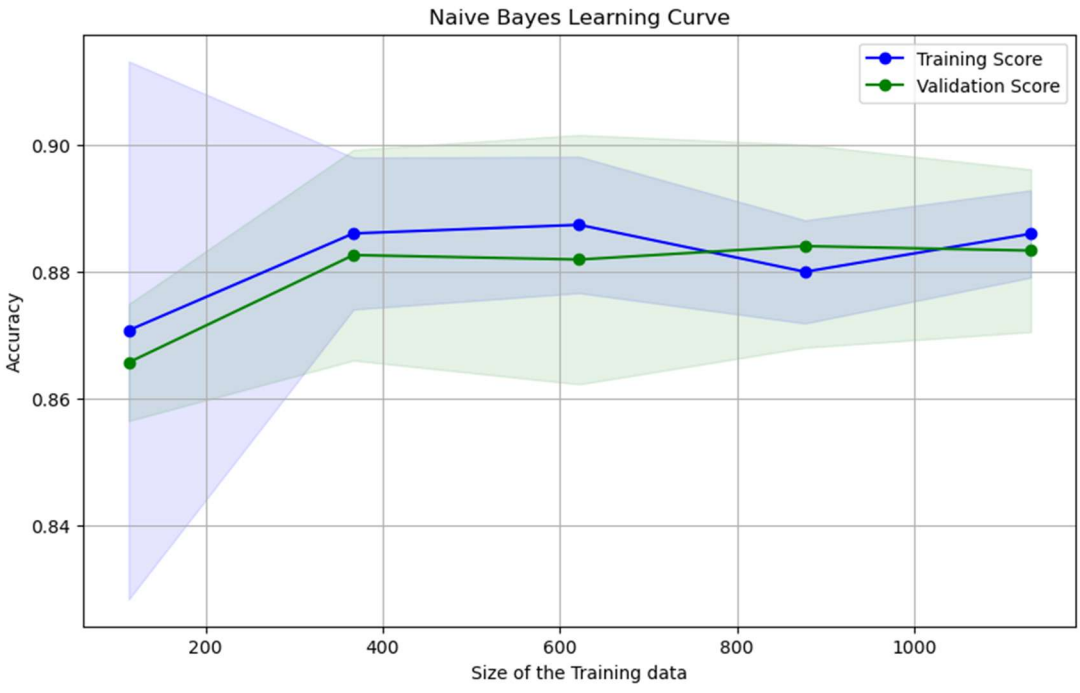
F1-Score: Ortalama %89, modelin genel performansını iyi bir seviyede tutmaktadır.

2. Confusion Matrix (Karışıklık Matrisi)



Model genellikle doğru sınıflandırma yapmaktadır, ancak sınıflar arası yakın özelliklere sahip veri noktalarında hata oranı yüksektir.

3. Learning Curve (Öğrenme Eğrisi)



Eğitim Skoru (Mavi Eğri):

Başlangıçta yüksek (%88 civarı) ve eğitim veri boyutu arttıkça sabit kalmaktadır.

Doğrulama Skoru (Yeşil Eğri):

Doğrulama skoru başlangıçta %86 civarındadır ve veri boyutuyla birlikte %88 civarına yükselerek sabitlenmektedir.

Eğriler Arasındaki Fark:

Eğitim ve doğrulama skorları arasındaki fark küçüktür ve eğitim veri boyutunun artmasıyla daha da azalmaktadır.

Bu, modelin overfitting yapmadığını ve genelleme performansının dengeli olduğunu göstermektedir.

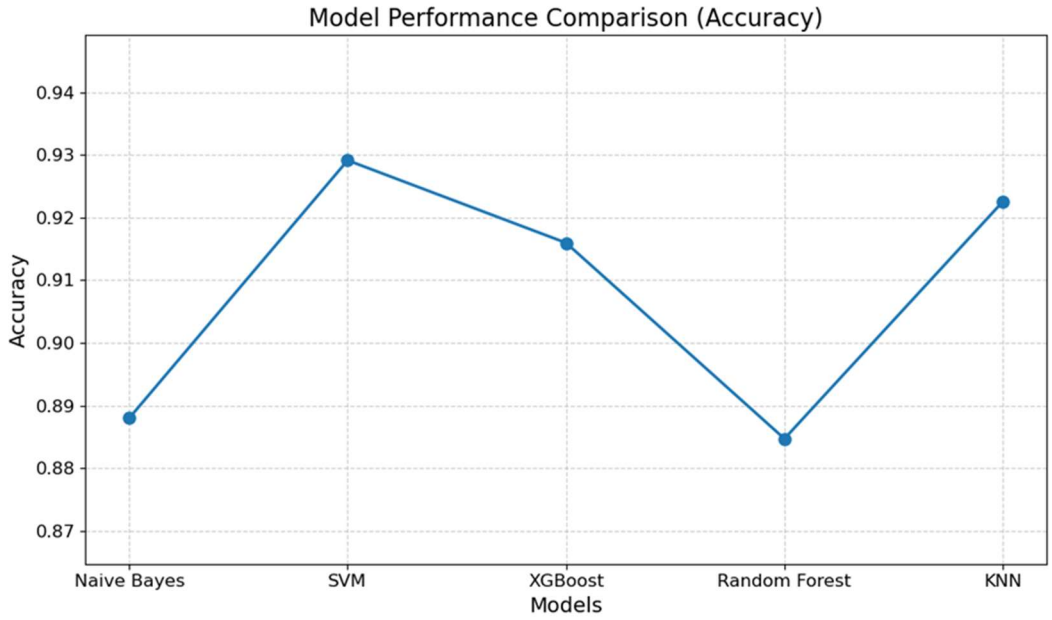
Genel Sonuçlar

- **Performans:** Naive Bayes modeli, %88.79 doğruluk oranı ile genelde dengeli bir performans sergilemektedir.
- **Hata Analizi:** Sınıf 1 ve sınıf 2 arasında yanlış tahminler dikkat çekmektedir. Bu sınıflar arasındaki ayrımı iyileştirmek için daha fazla özellik mühendisliği gerekebilir.
- **Genelleme Yeteneği:** Model overfitting yapmamış ve doğrulama performansı veri boyutuyla birlikte iyileşmiştir.

Bu analiz, Naive Bayes modelinin genel olarak dengeli bir performans sergilediğini ancak belirli sınıflar arasındaki karışıklıkların giderilmesi gerektiğini göstermektedir.

Model Performansları Karşılaştırması

Doğruluk Değerleri Karşılaştırması

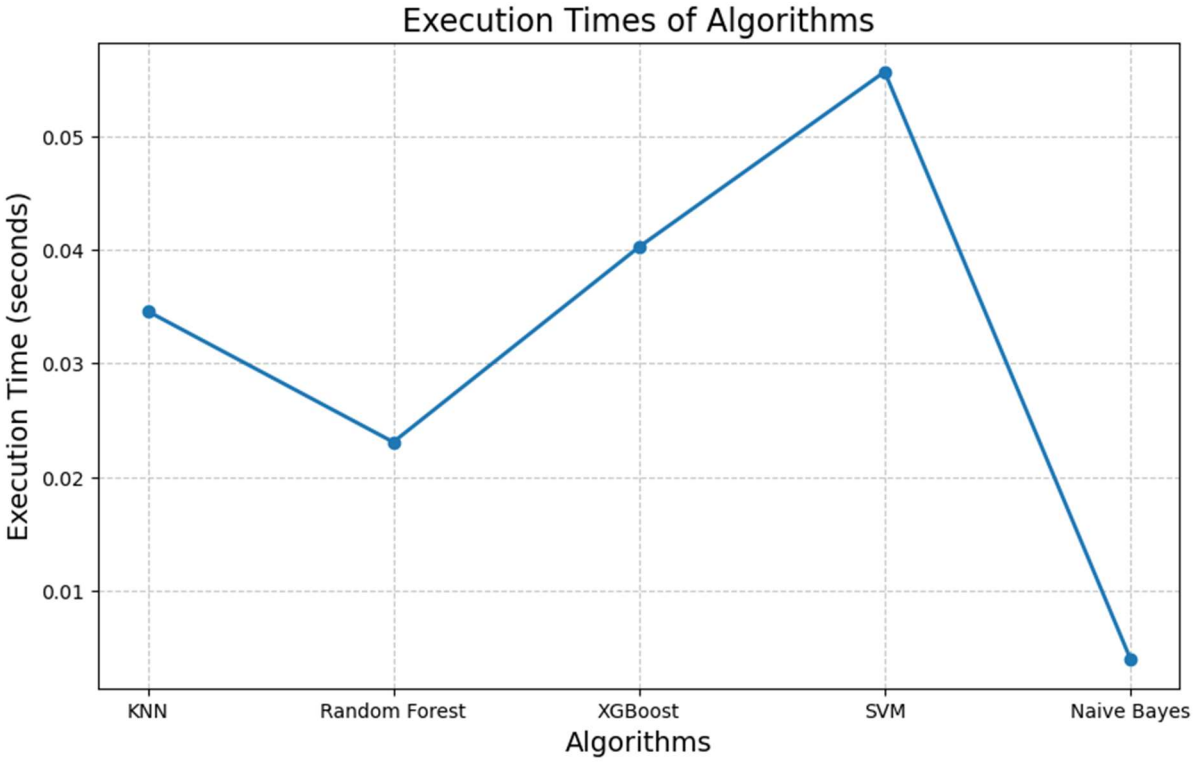


En yüksek doğruluk oranı: SVM, yaklaşık %93 doğruluk oranı ile en iyi performansı sergilemiştir. Bu, SVM'nin güçlü genelleme yeteneğini vurgulamaktadır.

Diğer yüksek performanslı algoritmalar: KNN (%92.26) ve XGBoost (%91.59) benzer şekilde yüksek doğruluk oranlarına sahiptir.

Random Forest ve Naive Bayes: Random Forest (%88.47) ve Naive Bayes (%88.79) diğer algoritmalara göre daha düşük doğruluk oranlarına sahiptir, ancak bu oranlar kabul edilebilir seviyededir. Model eğitimlerimiz genel olarak başarılı sonuçlanmıştır.

Çalışma Süreleri Karşılaştırması



En hızlı algoritma: Naive Bayes, çok kısa bir sürede çalışmaktadır. Bu, algoritmanın basit ve hesaplama açısından hafif bir yapıya sahip olduğunu göstermektedir.

En yavaş algoritma: SVM, diğer algoritmalara kıyasla daha fazla çalışma süresi gerektirmiştir. Bu durum, SVM'nin özellikle yüksek boyutlu veri setlerinde daha karmaşık hesaplamalar yapmasından kaynaklanabilir.

KNN, Random Forest ve XGBoost: Çalışma süreleri birbirine benzer seviyelerdedir. Bu algoritmalar, hesaplama açısından orta düzeyde yük oluşturmuşlardır.Sonuç olarak, Naive Bayes hız avantajı sunarken, SVM daha uzun sürede çalışmasına rağmen genelde yüksek performans sergilemektedir.