Bu veri seti, Eylül 2013'te Avrupa'daki kart sahiplerinin kredi kartı işlemlerini içeriyor. İki günlük bir dönemi kapsayan bu verilerde, 284.807 işlemden 492'si dolandırıcılık olarak işaretlenmiş, yani veri oldukça dengesizdir (sahtekarlık oranı %0.172). Veri seti yalnızca sayısal girdiler içeriyor ve çoğu değişken PCA (Ana Bileşen Analizi) ile dönüştürülmüş. 'V1'den 'V28'e kadar olan özellikler bu bileşenleri temsil ediyor. 'Time' ve 'Amount' özellikleri ise PCA ile dönüştürülmemiş orijinal özelliklerdir. 'Time', ilk işlem ile sonraki işlemler arasındaki zamanı, 'Amount' ise işlem tutarını ifade ediyor. Hedef değişken 'Class', dolandırıcılığı 1, diğer işlemleri ise 0 olarak sınıflandırıyor.

**Veri Setinin Genel Bilgileri:**

Veri setiniz 31 sütun ve 284,807 satır içeriyor. Sütunlar şunları içeriyor:

* Time: Her işlemin başlangıç işleminden itibaren geçen süre (saniye cinsinden).
* V1 ila V28: PCA dönüşümünden elde edilen 28 adet sayısal özellik.
* Amount: İşlem miktarı.
* Class: İşlemin dolandırıcılık olup olmadığını belirten etiket (0 = değil, 1 = evet).

import pandas as pd

# Veri setini yükleyin

df = pd.read\_csv('creditcard.csv')

# Veri setinin genel bilgilerini alın

df.info()

# Veri setinin özet istatistiklerini alın

df.describe()

**Veri Setinin Özet İstatistikleri:**

Veri setimizin özet istatistikleri aşağıdaki gibidir:

* **Time**:
  + Ortalama: 94,813.86 saniye
  + Standart Sapma: 47,488.15 saniye
  + Min: 0 saniye
  + Maks: 172,792 saniye
* **V1 - V28**: PCA ile dönüştürülmüş 28 özellik
  + Ortalama: Özelliklerin çoğu ortalama etrafında dağılmış, genellikle sıfıra yakın değerler
  + Standart Sapma: Özellikler arasında belirgin değişkenlik var
  + Min/Max: Özellikler geniş bir değer aralığında (örneğin, V2 -5.64 ile 2.20 arasında)
* **Amount**:
  + Ortalama: 88.35
  + Standart Sapma: 250.12
  + Min: 0
  + Maks: 25,691.16
* **Class**:
  + Ortalama: 0.0017 (Bu, veri setinde dolandırıcılık işlemlerinin oranının çok düşük olduğunu gösterir, yaklaşık %0.17)

Bu özet, veri setindeki bazı temel eğilimleri ve değişkenlikleri gösterir. Özellikle, Amount sütunundaki yüksek standart sapma, bazı işlemlerin oldukça yüksek miktarlarda olduğunu ve veri setinde önemli bir varyans bulunduğunu gösterir.

Birkaç grafik görelim.

import matplotlib.pyplot as plt

import seaborn as sns

# Sınıf (Fraud/No Fraud) dağılımı

plt.figure(figsize=(6, 4))

sns.countplot(x='Class', data=df, palette='viridis', hue='Class', legend=False)

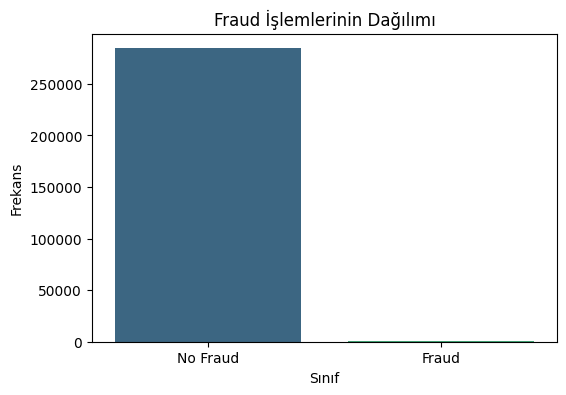
plt.title('Fraud İşlemlerinin Dağılımı')

plt.xlabel('Sınıf')

plt.ylabel('Frekans')

plt.xticks([0, 1], ['No Fraud', 'Fraud'])

plt.show()



# Zamanla işlem miktarları

plt.figure(figsize=(12, 6))

sns.scatterplot(x='Time', y='Amount', hue='Class', data=df, palette='coolwarm', alpha=0.5)

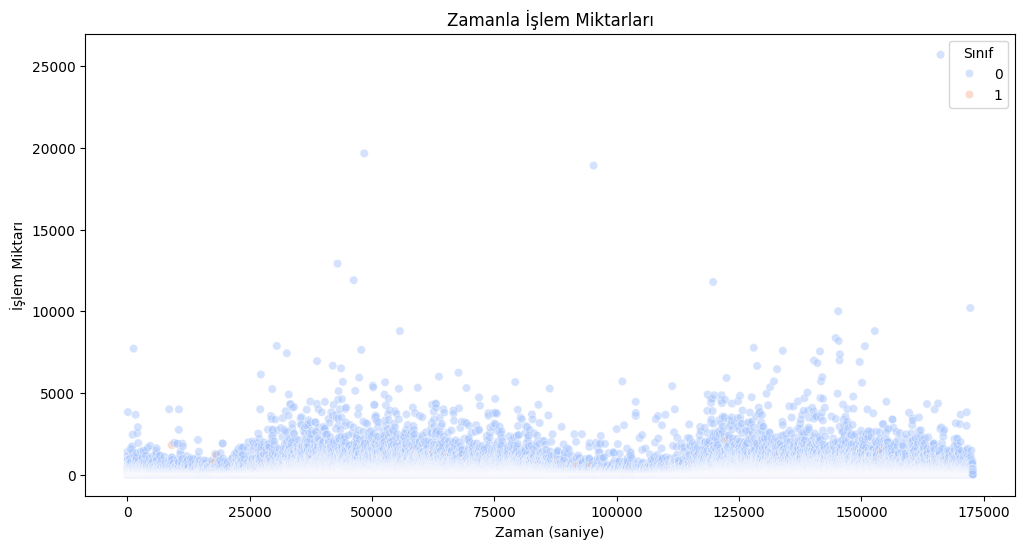
plt.title('Zamanla İşlem Miktarları')

plt.xlabel('Zaman (saniye)')

plt.ylabel('İşlem Miktarı')

plt.legend(title='Sınıf')

plt.show()



**5 - Veri Ön İşleme**

# Eksik değerleri kontrol et

print(df.isnull().sum())

Time 0

V1 0

V2 0

V3 0

V4 0

V5 0

V6 0

V7 0

V8 0

V9 0

V10 0

V11 0

V12 0

V13 0

V14 0

V15 0

V16 0

V17 0

V18 0

V19 0

V20 0

V21 0

V22 0

V23 0

V24 0

V25 0

V26 0

V27 0

V28 0

Amount 0

Class 0

dtype: int64

Veri setimizde eksik veri bulunmamaktadır. Yani, her bir sütunda eksik (NaN) değer yok. Bu iyi bir haber, çünkü eksik değerlerle başa çıkmak için ek bir adım atmanıza gerek kalmayacak.

**4. Keşifsel Veri Analizi (EDA)**

EDA aşamasında aşağıdaki adımları gerçekleştirebiliriz:

**a) Sınıf Dağılımını İnceleme**

Daha önce sınıf dağılımını gösteren bir **countplot** oluşturmuştuk. Dolandırıcılık işlemlerinin sayıca ne kadar az olduğunu görmüştük.

**b) Korelasyon Matrisi**

Özellikler arasındaki ilişkiyi görselleştirmek için bir korelasyon matrisi çıkarabiliriz. PCA ile dönüştürülmüş veriler olduğundan, korelasyonların nasıl dağıldığını görmek ilginç olabilir.

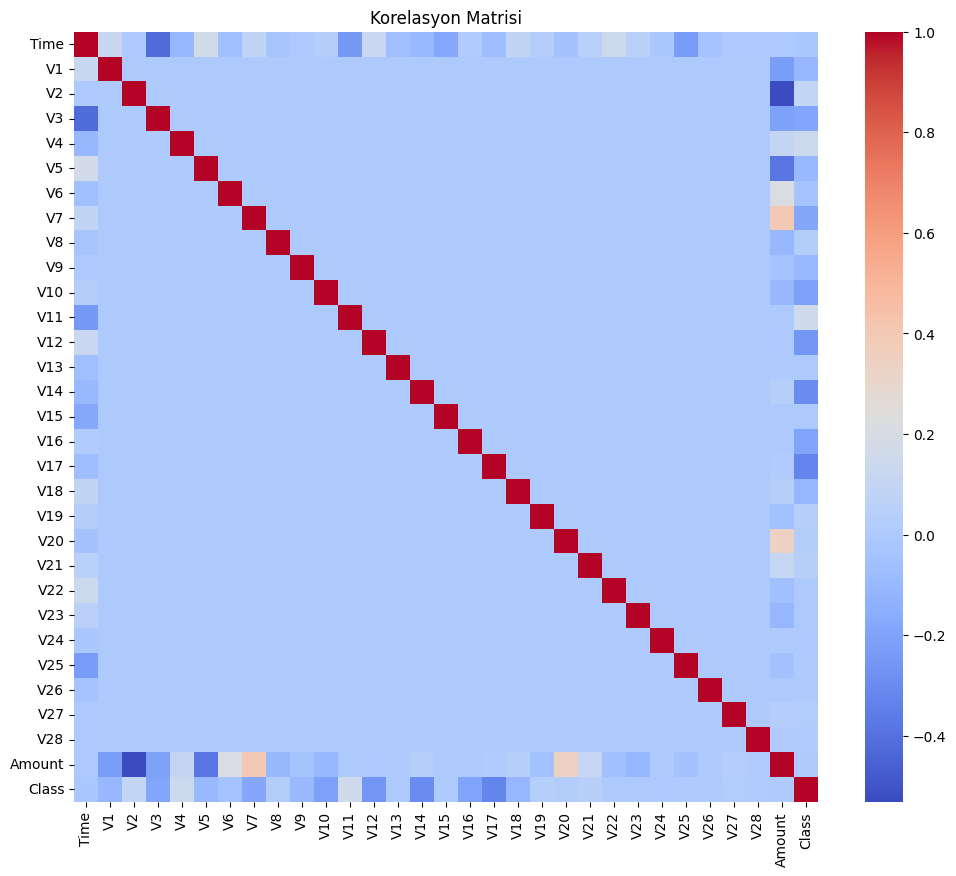
# Korelasyon matrisi

plt.figure(figsize=(12, 10))

sns.heatmap(df.corr(), cmap='coolwarm', annot=False, fmt=".2f")

plt.title('Korelasyon Matrisi')

plt.show()



Bu grafik, tüm değişkenlerin birbiriyle olan korelasyonlarını gösterir. Özellikle Amount ve Time değişkenlerinin dolandırıcılık ile olan ilişkisini inceleyebiliriz. Bu değişkenler arasındaki ilişki nötr yani güçsüzdür.

**c) İşlem Tutarlarının Dağılımı**

İşlem tutarlarının (Amount) nasıl dağıldığını görmek için bir dağılım grafiği çizebiliriz.

# İşlem tutarlarının dağılımı

plt.figure(figsize=(8, 6))

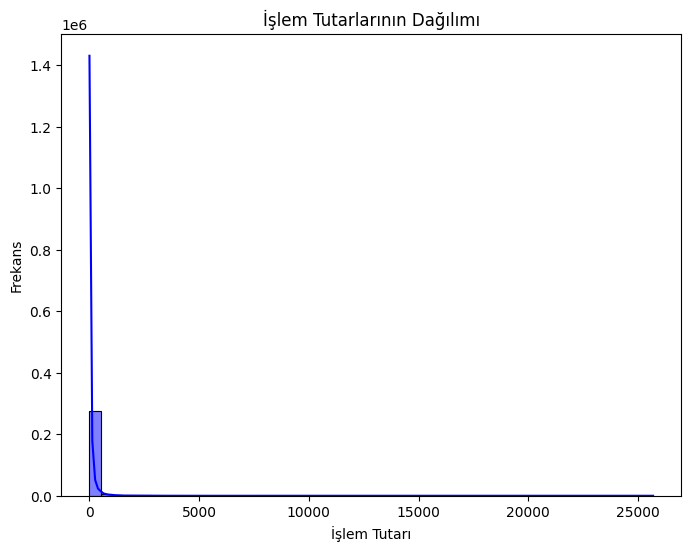
sns.histplot(df['Amount'], bins=50, kde=True, color='blue')

plt.title('İşlem Tutarlarının Dağılımı')

plt.xlabel('İşlem Tutarı')

plt.ylabel('Frekans')

plt.show()



Bu grafik, işlem tutarlarının çoğunun düşük olduğunu ve birkaç yüksek değer olduğunu gösterebilir. Bu dağılım, büyük olasılıkla dolandırıcılık işlemlerinin de yüksek tutarlı işlemlerle ilişkili olup olmadığını anlamak için detaylı analizlere ihtiyaç duyulacağını belirtebilir.

**d) Zamanla İşlem Sayısı**

İşlemlerin zaman içerisindeki dağılımını incelemek için bir çizgi grafiği kullanabiliriz.

# Zamanla işlem sayısı

plt.figure(figsize=(12, 6))

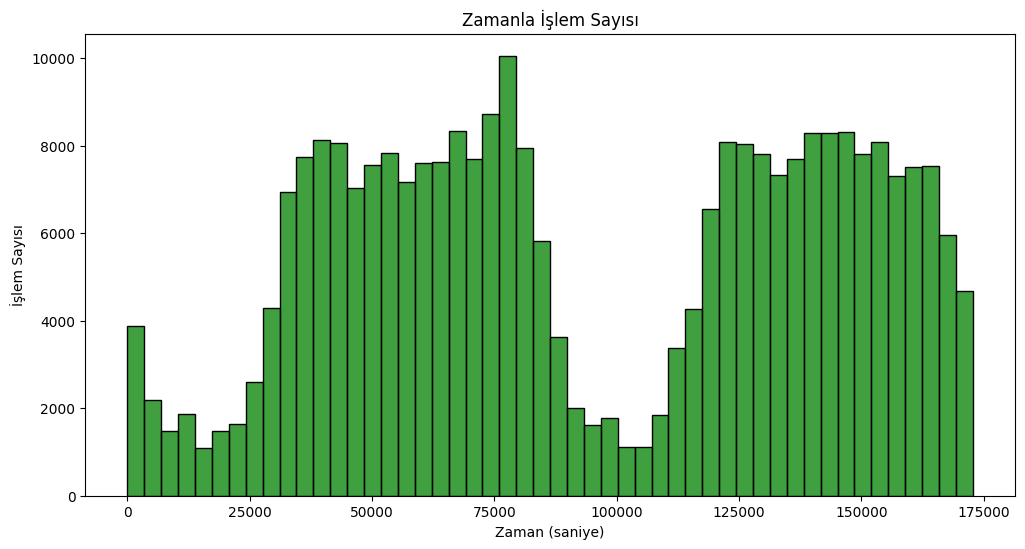
sns.histplot(df['Time'], bins=50, kde=False, color='green')

plt.title('Zamanla İşlem Sayısı')

plt.xlabel('Zaman (saniye)')

plt.ylabel('İşlem Sayısı')

plt.show()



Bu grafik, işlem sayısının zamanla nasıl değiştiğini ve belirli zaman aralıklarında daha yoğun olduğunu gösterebilir. Özellikle belirtilen zaman dilimlerinde yüksek işlem hacmi, belirli dönemlerde daha fazla işlem yapıldığını veya sistemde belirli zamanlarda yoğunluk yaşandığını belirtebilir. Bu bilgiyi dolandırıcılık analizinde kullanarak, dolandırıcılıkla ilişkili potansiyel zaman dilimlerini belirlemek için faydalı olabilir.

"5 - Veri Ön İşleme" aşamasına geldiğimizde, aşağıdaki adımları yapmamız gerekiyor. Daha önce eksik verileri kontrol ettik ve eksik veri olmadığını gördük, bu yüzden sonraki adımlar veri kümemizin özelliklerine göre uygulanmalı.

**Yapılacaklar:**

1. **Veri Temizleme**: Eksik veri olmadığını gördük, bu adımı atlayabiliriz.
2. **Özellik Normalizasyonu**: İşlem tutarı (Amount) ve zaman (Time) gibi özellikleri ölçeklendirmek önemli olabilir. Çünkü makine öğrenmesi modelleri özelliklerin ölçek farklılıklarından etkilenebilir. StandardScaler veya MinMaxScaler gibi bir ölçeklendirme tekniği kullanabiliriz.
3. **Label Encoding veya One-Hot Encoding**: Veri setimizde kategorik değişken yok, çünkü tüm veriler sayısal ve Class zaten etiketlenmiş bir sütun. Bu yüzden bu adımı da atlayabiliriz.
4. **Veri Kümesini Eğitim ve Test Olarak Bölme**: Eğitim ve test verilerine ayırarak modeli eğitmek ve değerlendirmek için train\_test\_split kullanacağız.

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

# Amount ve Time sütunlarını ölçeklendirme

scaler = StandardScaler()

df['scaled\_Amount'] = scaler.fit\_transform(df[['Amount']])

df['scaled\_Time'] = scaler.fit\_transform(df[['Time']])

# Ölçeklendirilmiş sütunlar eklendi, şimdi orijinal Amount ve Time sütunlarını kaldırabiliriz

df.drop(['Amount', 'Time'], axis=1, inplace=True)

# Veri setini bağımsız (X) ve bağımlı (y) değişkenlere ayırma

X = df.drop('Class', axis=1)

y = df['Class']

# Veri setini eğitim ve test setlerine ayıralım

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Eğitim ve test setlerinin boyutlarını kontrol edelim

print("Eğitim seti boyutu:", X\_train.shape)

print("Test seti boyutu:", X\_test.shape)

ÇIKTI

Eğitim seti boyutu: (227845, 30)

Test seti boyutu: (56962, 30)

Eğitim ve test setlerini başarıyla böldük. Eğitim setimiz 227,845 örnek içerirken, test setimiz 56,962 örnek içeriyor. Bu oran, modelimizin değerlendirme ve eğitim süreçleri için iyi bir denge sağlar.

**Devam Edilmesi Gereken Adımlar:**

1. **Model Seçimi ve Eğitimi:**
   * Veri kümemizi uygun makine öğrenmesi modelleriyle eğitmek için başlayabiliriz. Kredi kartı dolandırıcılığı tespitinde genellikle kullanılan modeller arasında Logistic Regression, Decision Trees, Random Forests, Gradient Boosting ve Neural Networks yer alır.
2. **Model Değerlendirme:**
   * Modelimizi eğittikten sonra, performansını değerlendirmek için çeşitli metrikler kullanabiliriz:
     + **AUC-ROC**: Modelinizin dolandırıcılığı ne kadar iyi tespit ettiğini gösterir.
     + **Confusion Matrix**: Doğru ve yanlış tahminlerinizi incelememizi sağlar.
     + **Precision, Recall, F1-Score**: Sınıflandırma performansınızı detaylı bir şekilde değerlendirir.
3. **Hiperparametre Ayarlama:**
   * Modelinizin performansını artırmak için hiperparametre optimizasyonu yapabiliriz. Grid Search veya Random Search yöntemlerini kullanarak en iyi hiperparametreleri bulabiliriz.
4. **Sonuçları Yorumlama ve Sunma:**
   * Sonuçları yorumlayın ve bulgularınızı raporlayın. Özellikle dolandırıcılık işlemlerini doğru tespit etme oranımızı ve modelimizin hangi alanlarda güçlü veya zayıf olduğunu belirtin.

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix, roc\_auc\_score

# Modeli oluşturma ve eğitme

model = RandomForestClassifier(n\_estimators=20, random\_state=42)

model.fit(X\_train, y\_train)

# Tahminler

y\_pred = model.predict(X\_test)

y\_proba = model.predict\_proba(X\_test)[:, 1]

# Performans değerlendirmesi

cm = confusion\_matrix(y\_test, y\_pred)

cr = classification\_report(y\_test, y\_pred)

auc = roc\_auc\_score(y\_test, y\_proba)

# Sonuçları yazdırma

print(f"Confusion Matrix:\n{cm}\n")

print(f"Classification Report:\n{cr}\n")

print(f"AUC-ROC Score: {auc:.4f}")

ÇIKTI

Confusion Matrix:

[[56859 5]

[ 23 75]]

Classification Report:

precision recall f1-score support

0 1.00 1.00 1.00 56864

1 0.94 0.77 0.84 98

accuracy 1.00 56962

macro avg 0.97 0.88 0.92 56962

weighted avg 1.00 1.00 1.00 56962

AUC-ROC Score: 0.9332

**True Negatives (TN)**: 56,859 (Dolandırıcılık olmayan işlemler doğru şekilde tahmin edildi)

**False Positives (FP)**: 5 (Dolandırıcılık olmayan işlemler yanlışlıkla dolandırıcılık olarak tahmin edildi)

**False Negatives (FN)**: 23 (Dolandırıcılık olan işlemler yanlışlıkla dolandırıcılık olmayan olarak tahmin edildi)

**True Positives (TP)**: 75 (Dolandırıcılık olan işlemler doğru şekilde tahmin edildi)

**Classification Report**

* **Precision**: Dolandırıcılık olarak tahmin edilen işlemlerin ne kadarının gerçekten dolandırıcılık olduğunu gösterir. Dolandırıcılık olan işlemler için 0.94, dolandırıcılık olmayan işlemler için 1.00.
* **Recall**: Gerçek dolandırıcılık işlemlerinin ne kadarının doğru tahmin edildiğini gösterir. Dolandırıcılık olan işlemler için 0.77, dolandırıcılık olmayan işlemler için 1.00.
* **F1-Score**: Precision ve Recall'ün harmonik ortalamasıdır. Dolandırıcılık olan işlemler için 0.84, dolandırıcılık olmayan işlemler için 1.00.
* **Accuracy**: Modelin genel doğruluğunu gösterir. Toplamda %100'e yakın bir doğruluk elde edilmiş.

**AUC-ROC Score**

* **AUC-ROC Score**: 0.9332. Bu skor, modelin genel performansını ve dolandırıcılık işlemlerini doğru tahmin etme yeteneğini gösterir. 1'e yakın bir değer, modelin iyi performans gösterdiğini belirtir.

**Genel Yorumlar**

* Model oldukça yüksek bir doğruluk oranına sahip, ancak dolandırıcılık olan işlemler için recall değerinin daha düşük olduğunu gözlemleyebilirsiniz (0.77). Bu, modelin dolandırıcılık işlemlerini tespit etme yeteneğinin sınırlı olduğunu gösterir. Dolandırıcılık işlemleri genellikle nadir olaylardır ve bu nedenle, daha iyi bir model geliştirmek için daha fazla özellik mühendisliği veya hiperparametre optimizasyonu gerekebilir.
* False Negatives sayısının (23) daha düşük olması, modelin dolandırıcılık işlemlerini tanımlama yeteneğinin geliştirilmesi gerektiğini işaret eder.

Eğer daha iyi sonuçlar elde etmek istiyorsak, modelin farklı hiperparametrelerle veya farklı algoritmalarla yeniden eğitilmesi gerekebilir. Ayrıca, veri dengesizliğini ele almak için çeşitli stratejiler (örneğin SMOTE gibi) uygulamak da faydalı olabilir.

**Model Seçimi ve Çapraz Doğrulama**

Farklı modelleri deneyip çapraz doğrulama ile performanslarını değerlendirebiliriz.

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

# Modelleri tanımla

models = {

    'Logistic Regression': LogisticRegression(),

    'K-Nearest Neighbors': KNeighborsClassifier(),

    'Support Vector Machine': SVC(probability=True)

}

# Çapraz doğrulama ile modelleri değerlendir

for name, model in models.items():

    scores = cross\_val\_score(model, X\_train, y\_train, cv=3, scoring='roc\_auc')

    print(f"{name} AUC-ROC: {scores.mean():.4f} ± {scores.std():.4f}")

ÇIKTI

Logistic Regression AUC-ROC: 0.9756 ± 0.0023

K-Nearest Neighbors AUC-ROC: 0.9136 ± 0.0099

Support Vector Machine AUC-ROC: 0.9366 ± 0.0173

**Algoritma Seçimi & Hiperparametre Optimizasyonu İçin Adımlar:**

1. **Model Seçimi:**
   * **En yüksek AUC-ROC skoru** Logistic Regression modeline ait (0.9756). Bu, modelimizin en iyi performansı gösterdiğini ve diğer iki modelden (KNN ve SVM) daha iyi olduğunu gösteriyor.
2. **Hiperparametre Optimizasyonu:**
   * Şimdi Logistic Regression modelinin hiperparametrelerini optimize edebiliriz. Bu, Grid Search veya Randomized Search gibi yöntemlerle yapılabilir.

Örnek Hiperparametre Optimizasyonu (Grid Search) için Kod:

from sklearn.model\_selection import GridSearchCV

# Hiperparametre ızgarası

param\_grid = {

'C': [0.001, 0.01, 0.1, 1, 10, 100],

'solver': ['liblinear', 'saga']

}

# Grid Search

grid\_search = GridSearchCV(LogisticRegression(max\_iter=500

), param\_grid, cv=3, scoring='roc\_auc', n\_jobs=-1)

grid\_search.fit(X\_train, y\_train)

print(f"En İyi Parametreler: {grid\_search.best\_params\_}")

print(f"En İyi AUC-ROC: {grid\_search.best\_score\_:.4f}")

ÇIKTI

En İyi Parametreler: {'C': 0.01, 'solver': 'saga'}

En İyi AUC-ROC: 0.9823

Bu çıktı, **Lojistik Regresyon** modelin için yapılan Grid Search sonucunda en iyi hiperparametre kombinasyonunu ve bu kombinasyonla elde edilen **AUC-ROC** skorunu gösteriyor.

**Yorum:**

* **En İyi Parametreler: {'C': 0.01, 'solver': 'saga'}**:
  + **C** parametresi, Lojistik Regresyonda cezalandırma terimini kontrol eden bir hiperparametredir. Küçük bir **C** değeri (0.01) modelin daha güçlü bir şekilde düzenleme (regularization) yaptığı anlamına gelir, bu da overfitting'i önlemeye yardımcı olabilir.
  + **solver: 'saga'** ise Lojistik Regresyon için kullanılan optimizasyon algoritmasıdır. **'saga'** büyük veri kümeleri ve L1/L2 regularization ile çalışmak için uygun bir yöntemdir.
* **En İyi AUC-ROC: 0.9823**: Bu değer, modelin performansını ölçen bir metriktir. 0.9823, modelin sınıflandırma problemini çözmedeki doğruluğunun oldukça yüksek olduğunu gösterir. Bu AUC-ROC skoruyla, model hem pozitif hem de negatif sınıfları oldukça başarılı bir şekilde ayırıyor demektir.

**Sonuç:**

Bu hiperparametrelerle Lojistik Regresyon modelin oldukça iyi bir performans gösteriyor. Bundan sonra, modelin bu hiperparametrelerle test verisi üzerindeki performansını değerlendirebiliriz ve diğer modellerle karşılaştırabiliriz.

Eğitim aşamasında en iyi hiperparametreler belirlendiğine göre, şimdi test verisine geçelim ve modelin AUC-ROC skorunu ölçelim.

from sklearn.metrics import roc\_auc\_score

# En iyi modeli test verisi ile değerlendir

best\_model = grid\_search.best\_estimator\_

y\_pred\_prob = best\_model.predict\_proba(X\_test)[:, 1] # Pozitif sınıfın olasılıklarını alıyoruz

# Test seti üzerinde AUC-ROC skoru

auc\_roc\_test = roc\_auc\_score(y\_test, y\_pred\_prob)

print(f"Lojistik Regresyon Test AUC-ROC: {auc\_roc\_test:.4f}")

ÇIKTI

Lojistik Regresyon Test AUC-ROC: 0.9809

Lojistik Regresyon modelimizin test AUC-ROC skoru 0.9809, bu oldukça iyi bir sonuç Sonuçlara göre hiperparametre optimizasyonu yapılmış Lojistik Regresyon modeliyle devam etmemiz uygun görünüyor.

Gözetimli Öğrenme Algoritmalarından, Lojistik Regresyon modelini seçtik ve Grid Search yöntemi ile hiperparametre optimizasyonu yaptık. Özellikle modelin C ve solver parametreleri için bir hiperparametre aralığı belirledik. Yaptığımız optimizasyon sonucunda en iyi parametreler {C: 0.01, solver: 'saga'} olarak bulundu ve modelin AUC-ROC performansı optimize edildi. Diğer modellerden daha iyi performans gösterdiği için bu modelle devam etmeye karar verdik.

7 - Model Değerlendirme

* **Confusion Matrix**:
  + Gerçek negatif (0) ve pozitif (1) sınıflandırmalarda model çok yüksek doğruluk gösteriyor. Yani, 56855 doğru negatif ve 54 doğru pozitif sınıflama yapmış.
  + Modelin 9 yanlış negatif ve 44 yanlış pozitif sınıflaması var.
* **Classification Report**:
  + **Doğruluk (Accuracy)**: %100 (1.00), genel olarak modelin ne kadar doğru sınıflandırdığını gösterir.
  + **Kesinlik (Precision)**: Pozitif sınıflar için %86 (0.86). Pozitif olarak sınıflandırılan örneklerin ne kadarının gerçekten pozitif olduğunu gösterir.
  + **Duyarlılık (Recall)**: Pozitif sınıflar için %55 (0.55). Gerçek pozitif örneklerin ne kadarının doğru bir şekilde pozitif olarak sınıflandırıldığını gösterir.
  + **F1 Puanı (F1 Score)**: Pozitif sınıflar için %67 (0.67). Kesinlik ve duyarlılığın harmonik ortalamasıdır.
* **AUC-ROC Score**: 0.9809, modelin genel performansını iyi bir şekilde yansıtıyor. AUC-ROC skoru yüksek olduğunda, modelin sınıflandırma başarısı yüksektir.

Bu sonuçlar, modelimizin genel performansının oldukça iyi olduğunu ve doğru sınıflandırmalar gerçekleştirdiğini gösteriyor. Özellikle yüksek doğruluk, iyi AUC-ROC skoru ve detaylı raporlar, optimize edilmiş Lojistik Regresyon modelinin uygun olduğunu gösteriyor.