BURSA TEKNİK ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

BLM0463 VERİ MADENCİLİĞİNE GİRİŞ

WHOLESALE CUSTOMERS DATASET – DECİSİON TREES

21360859044

SIDIKA FİRAT

2024-2025

İçindekiler

1. Veri Seti Tanımı

1.1 Veri Seti Adı ve Kaynağı

- **Veri Seti Adı:** Wholesale Customers Data Set
- **Kaynak:** UCI Machine Learning Repository
- **Link:** https://archive.ics.uci.edu/dataset/292/wholesale+customers
- **Açıklama:** Bu veri seti, toptancı müşterilerinin farklı ürün kategorilerindeki yıllık harcama bilgilerini içermektedir.

1.2 Veri Seti Tanımı: Wholesale Customers Dataset

Wholesale Customers veri seti, bir toptan dağıtım yapan firmanın müşterilerine ait yıllık harcama verilerini içermektedir. Veri seti, her bir müşterinin çeşitli ürün kategorilerine yaptığı harcamaları göstermektedir ve özellikle sınıflandırma (classification) ve kümeleme (clustering) gibi makine öğrenmesi görevleri için kullanılmak üzere hazırlanmıştır.
Veri seti, UCI Machine Learning Repository tarafından sağlanmıştır ve Margarida Cardoso tarafından 2013 yılında bağışlanmıştır. Veri seti 440 müşteriye ait gözlem içermektedir ve her bir müşteri için 8 değişken (özellik) bulunmaktadır.

1.3 Veri Seti Özellikleri

- **Gözlem Sayısı:** 440
- **Özellik Sayısı:** 8
- **Veri Türü:** Sayısal (integer) ve kategorik (nominal)
- **Eksik Değer:** Yok
- **Konu Alanı:** İş dünyası (Business)
- **Kullanım Alanları:** Sınıflandırma, Kümeleme

1.4 Özniteliklerin Açıklaması

Sütun Adı	Veri Tipi	Açıklama	Ölçüm Birimi
Channel	Kategorik	Müşteri kanalı (1: Hotel/Restaurant/Cafe, 2: Retail)	-
Region	Kategorik	Müşterinin bulunduğu bölge (1: Lizbon, 2: Porto, 3: Diğer bölgeler)	-
Fresh	Sayısal	Taze ürünler (örneğin meyve, sebze) harcaması	Para birimi
Milk	Sayısal	Süt ürünleri harcaması	Para birimi
Grocery	Sayısal	Bakkal ürünleri harcaması	Para birimi
Frozen	Sayısal	Dondurulmuş ürünler harcaması	Para birimi
Detergents_Paper	Sayısal	Temizlik kağıtları ve deterjan harcaması	Para birimi
Delicassen	Sayısal	Şarküteri ürünleri harcaması	Para birimi

1.5 Dağılım

- **Region Dağılımı:**
  - Lisbon: 77
  - Oporto: 47
  - Other: 316
- **Channel Dağılımı:**
  - Horeca: 298
  - Retail: 142

1.6 İstatiksel Bilgiler

- `Fresh`: Min: 3, Max: 112151, Ortalama: 12000.30, Std: 12647.33
- `Milk`: Min: 55, Max: 73498, Ortalama: 5796.27, Std: 7380.38
- `Grocery`: Min: 3, Max: 92780, Ortalama: 7951.28, Std: 9503.16
- `Frozen`: Min: 25, Max: 60869, Ortalama: 3071.93, Std: 4854.67
- `Detergents_Paper`: Min: 3, Max: 40827, Ortalama: 2881.49, Std: 4767.85
- `Delicassen`: Min: 3, Max: 47943, Ortalama: 1524.87, Std: 2820.11

1.7 Veri Setinin Amacı

Wholesale Customers veri seti, bir toptan satış yapan firmanın müşteri segmentlerini anlamak, sınıflandırmak ve potansiyel iş stratejileri geliştirmek amacıyla hazırlanmıştır. Veri setinin temel amacı, müşterilerin yıllık harcamalarına göre hangi **kanala** (Horeca veya Retail) ait olduklarını tahmin etmek ya da bu müşterileri benzer harcama davranışlarına göre **gruplamaktır**.

Veri seti sayesinde işletmeler:

- Farklı müşteri gruplarının (kanal/bölge) **satın alma alışkanlıklarını** analiz edebilir,
- Satış ve pazarlama stratejilerini **müşteri segmentlerine göre özelleştirebilir**,
- Müşteri davranışlarını temel alarak **hedefli kampanyalar** düzenleyebilir,
- Yeni müşterilerin hangi kategoriye ait olacağını **sınıflandırma algoritmalarıyla tahmin edebilir**.

Bu çalışma kapsamında veri seti, özellikle **Decision Tree (Karar Ağacı) algoritması** ile müşterilerin **Channel (Horeca veya Retail)** sınıfına ait olup olmadığını tahmin etmek amacıyla kullanılmıştır. Amaç, eğitim verisinden öğrenilen örüntülere dayanarak yeni müşteri örneklerinin doğru bir şekilde sınıflandırılmasını sağlamaktır.

1.8 Veri Setinin Önemi

Bu veri seti, toptan satış yapan bir şirketin müşterilerinin yıllık harcamalarını içeriyor. Hangi bölgede oldukları ve hangi satış kanalını kullandıkları da belirtilmiş. Bu sayede:

- Müşteriler daha iyi tanınabilir, gruplandırılabilir.
- Şirketler kime, ne satacaklarına daha doğru karar verebilir.
- Satış ve pazarlama stratejileri daha iyi planlanabilir.
- Hem sınıflandırma hem de kümeleme gibi yapay zeka uygulamaları için uygundur.
- Gerçek hayattan alındığı için hem derslerde hem de projelerde rahatlıkla kullanılabilir.

2. Decision Tree Tanımı

Karar ağacı, tıpkı bir soru-cevap oyunu gibi çalışan bir makine öğrenmesi yöntemidir. Bir problemi adım adım sorularla bölerek çözer. Her bir soru, veriyle ilgili bir özelliğe (örneğin: süt harcaması ne kadar?) dayanır. Sorunun cevabına göre başka bir soruya geçilir, tıpkı dallara ayrılan bir ağaç gibi. En sonunda ağacın yapraklarına ulaştığımızda, kararımızı vermiş oluruz: örneğin bir müşteri "perakende mi yoksa horeca mı" gibi.

Neden kullanılır?

- Yorumlaması kolaydır.
- Görsel olarak da anlaşılır.
- Hem sınıflandırma (hangi sınıfa ait?) hem de regresyon (ne kadar?) problemlerinde kullanılabilir.

Özetle: Karar ağacı, bir karar vermek için veriye "akıllı" sorular sorarak ilerleyen bir yöntemdir.

3. Veri Setinin İşlenmesi

3.1 Veri Setini Yükleme

```
from google.colab import files
uploaded = files.upload()
```

3.2 Veriyi Okuma

```
[ ]  import pandas as pd

     # Doğru ayırıcı ile oku
     df = pd.read_csv("Wholesale customers data.csv", sep=";")

     # Sütun isimlerini kontrol et
     print(df.columns.tolist())
```

```
['Channel', 'Region', 'Fresh', 'Milk', 'Grocery', 'Frozen', 'Detergents_Paper', 'Delicassen']
```

3.3 Özellik(x) ve Hedef Değişken(y) Ayırma:

Bu projede hedef değişken (tahmin etmeye çalıştığımız şey) Channel sütunudur. Diğer sütunlar modelin tahmin yaparken kullanacağı özelliklerdir.

```
X = df.drop("Channel", axis=1)
y = df["Channel"]
```

3.4 Model Kurma ve Eğitme

Bu çalışmada, veri setimi önce eğitim ve test olarak ayırdım (%80 eğitim, %20 test). Ardından karar ağacı algoritmasını kullanarak modelimi eğittim ve test verileriyle tahmin yaparak doğruluk oranını hesapladım. Bu sayede modelimin ne kadar başarılı olduğunu görmüş oldum.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Veriyi eğitim ve test olarak ayır (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Decision Tree modelini oluştur ve eğit
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Tahmin yap ve doğruluğu hesapla
y_pred = model.predict(X_test)
print("Doğruluk Skoru:", accuracy_score(y_test, y_pred))
```

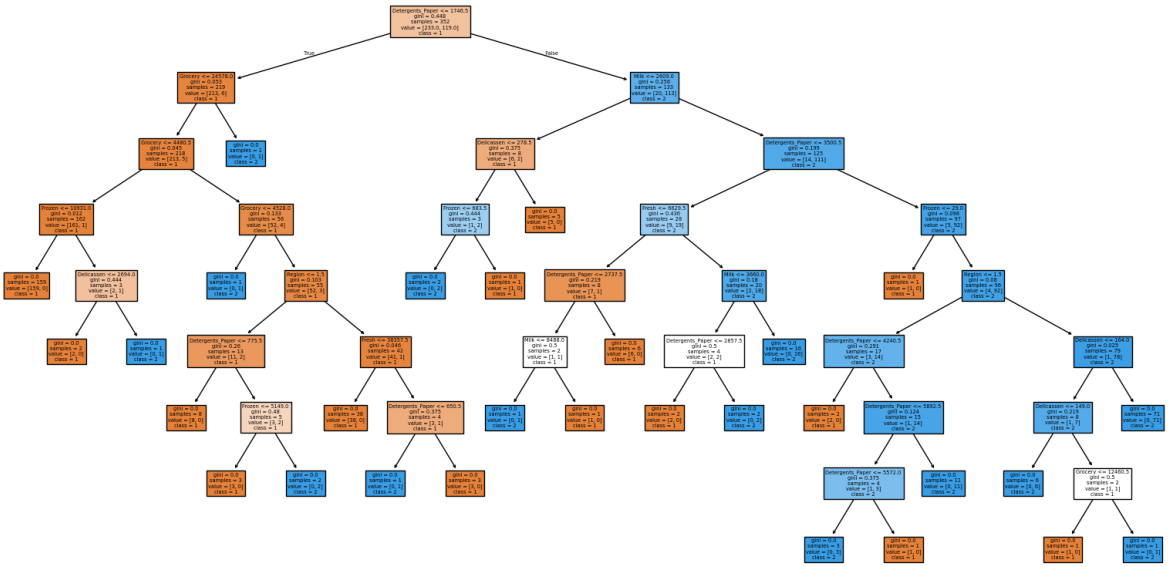
```
Doğruluk Skoru: 0.875
```

3.5 Decision Tree Görselleştirme

Bu grafik:

- Hangi özelliklerin önemli olduğunu
- Hangi koşullarla sınıflandırma yaptığını
- Hangi sınıfa (örneğin Channel 1 veya 2) daha çok örnek düştüğünü gösterecek.

```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

plt.figure(figsize=(20,10))
plot_tree(model,
          feature_names=X.columns,
          class_names=[str(cls) for cls in model.classes_],
          filled=True)
plt.show()
```



4. Modeli İyileştirme

4.1 Overfitting Engelleme

Bu adımda karar ağacımın karmaşıklığını kontrol etmek için max_depth=3 parametresini belirledim. Böylece ağacın derinliğini sınırlayarak aşırı öğrenmeyi (overfitting) engellemeyi amaçladım. Bu sayede model daha sade ve genellenebilir hale geldi.

```
[ ]  model = DecisionTreeClassifier(max_depth=3)
```

4.2 Performans İyileştirme

Bu aşamada, oluşturduğum karar ağacı modelini eğitim verileriyle eğittim ve ardından test verileri üzerinde tahminler yaptım. Accuracy_score fonksiyonu ile modelin doğruluk oranını hesapladım. Daha önceki modele kıyasla doğruluk skorunun arttığını gözlemledim, bu da yaptığım ayarların modelin performansını iyileştirdiğini gösteriyor.

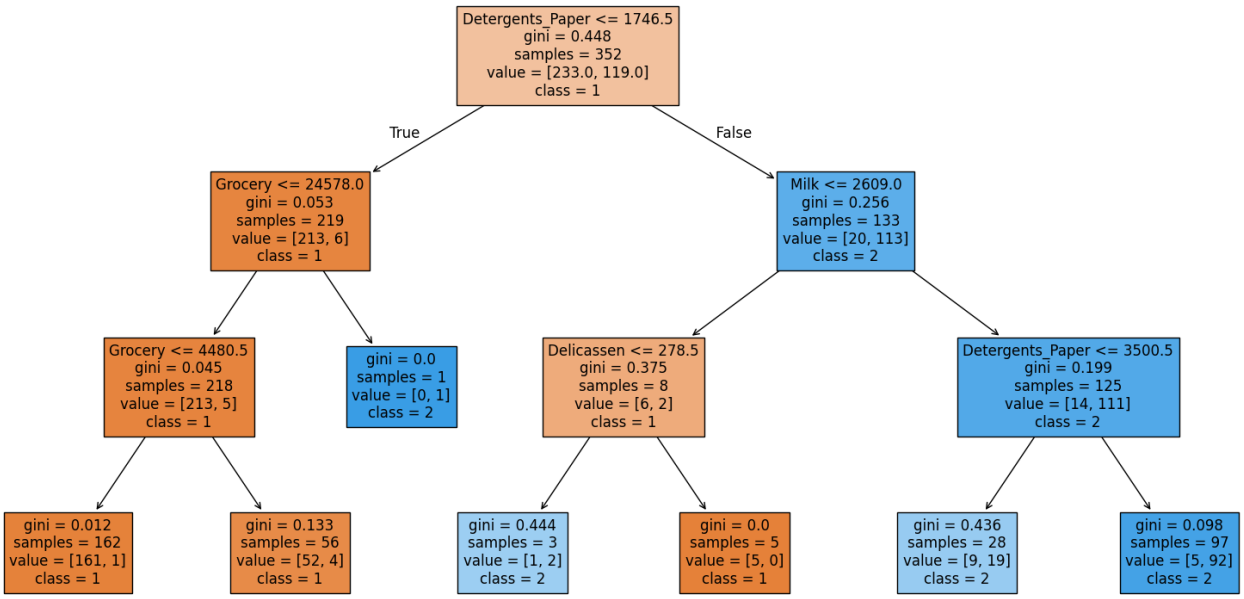
```
[ ]  # Modeli eğit
     model.fit(X_train, y_train)

     # Test verisiyle tahmin yap
     y_pred = model.predict(X_test)

     # Başarı oranını yazdır
     from sklearn.metrics import accuracy_score
     print("Doğruluk Skoru:", accuracy_score(y_test, y_pred))
```

```
Doğruluk Skoru: 0.8863636363636364
```

4.3 Karar Ağacı

Bu bölümde karar ağacı modelimin görselleştirmesini yaptım. plot_tree fonksiyonu sayesinde modelin nasıl karar verdiğini, hangi özelliklere göre dallandığını ve hangi sınıflara ayırdığını grafiksel olarak görebildim. Bu görsel, modelin nasıl çalıştığını daha iyi anlamamı sağladı.

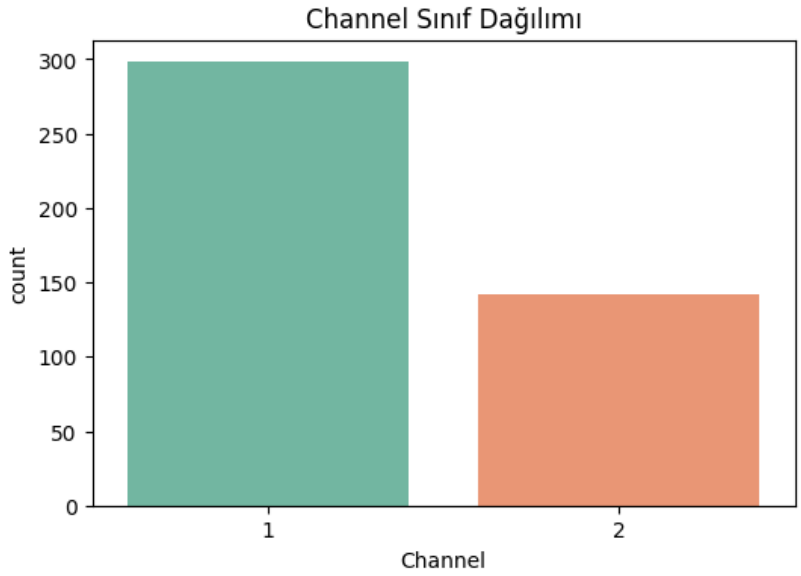
```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

plt.figure(figsize=(20,10))
plot_tree(model,
          feature_names=X.columns,
          class_names=[str(cls) for cls in model.classes_],
          filled=True)
plt.show()
```



4.4 Sınıf Dağılımlarının Görselleştirilmesi

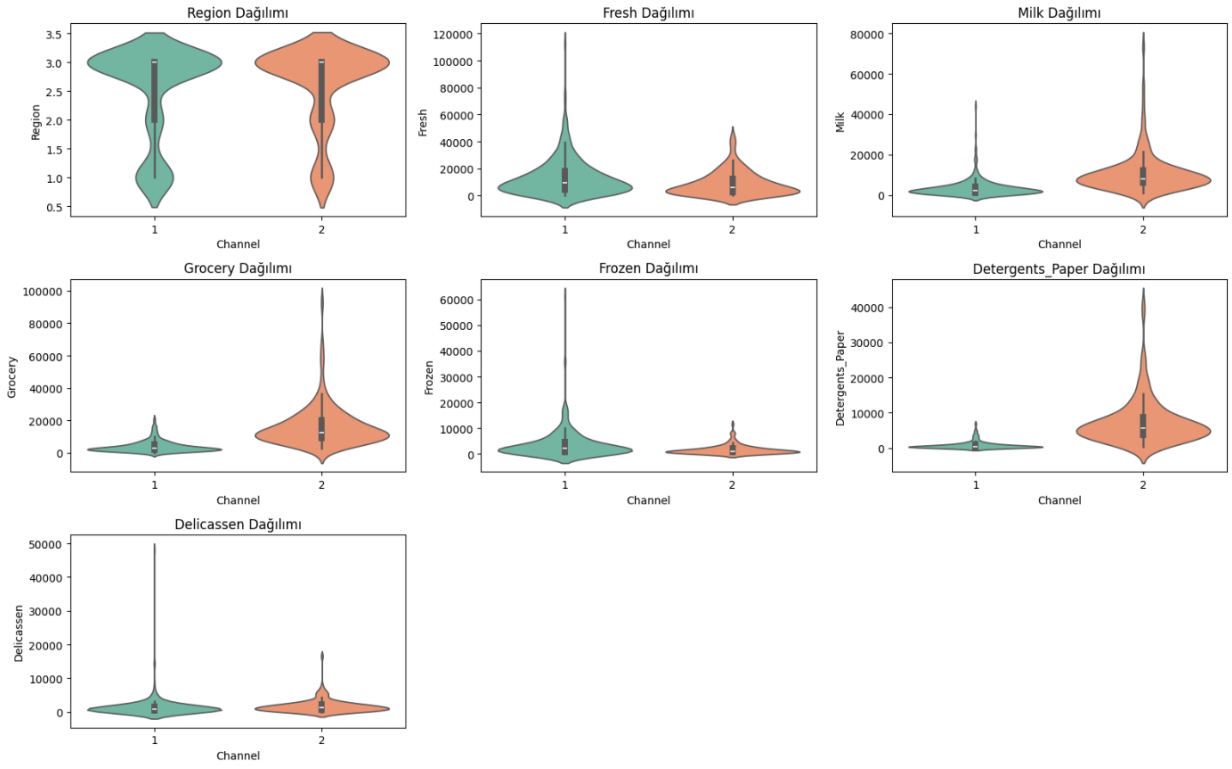
Her bar, farklı bir sınıfın veri sayısını temsil ediyor. Renkler sınıfları ayırt etmek için kullanıldı.

```
plt.figure(figsize=(6,4))
sns.countplot(x='Channel', data=df, palette='Set2')
plt.title('Channel Sınıf Dağılımı')
plt.show()
```



4.5 Özniteliklerin Sınıflarına Göre Dağılımlarını Gösteren Violin Grafikleri

Her bir özelliğin Channel sınıflarına göre dağılımını violin plot ile görselleştirdim. Bu grafikler sayesinde sınıflar arasındaki veri yoğunluklarını ve farklılıkları daha net görebiliyorum.



4.6 Öznitelik Önemlerinin Analizi

Bu kısımda modelimin hangi özelliklere ne kadar önem verdiğini inceledim. Detergents_Paper özelliği %91 gibi çok yüksek bir oranla modelin kararlarında en etkili faktör olmuş. Ardından sırasıyla Milk, Grocery ve Delicassen özellikleri gelmiş. Region, Fresh ve Frozen ise bu modelde neredeyse hiç etkili olmamış. Bu analiz, modelin neye göre tahmin yaptığını anlamamda yardımcı oldu.

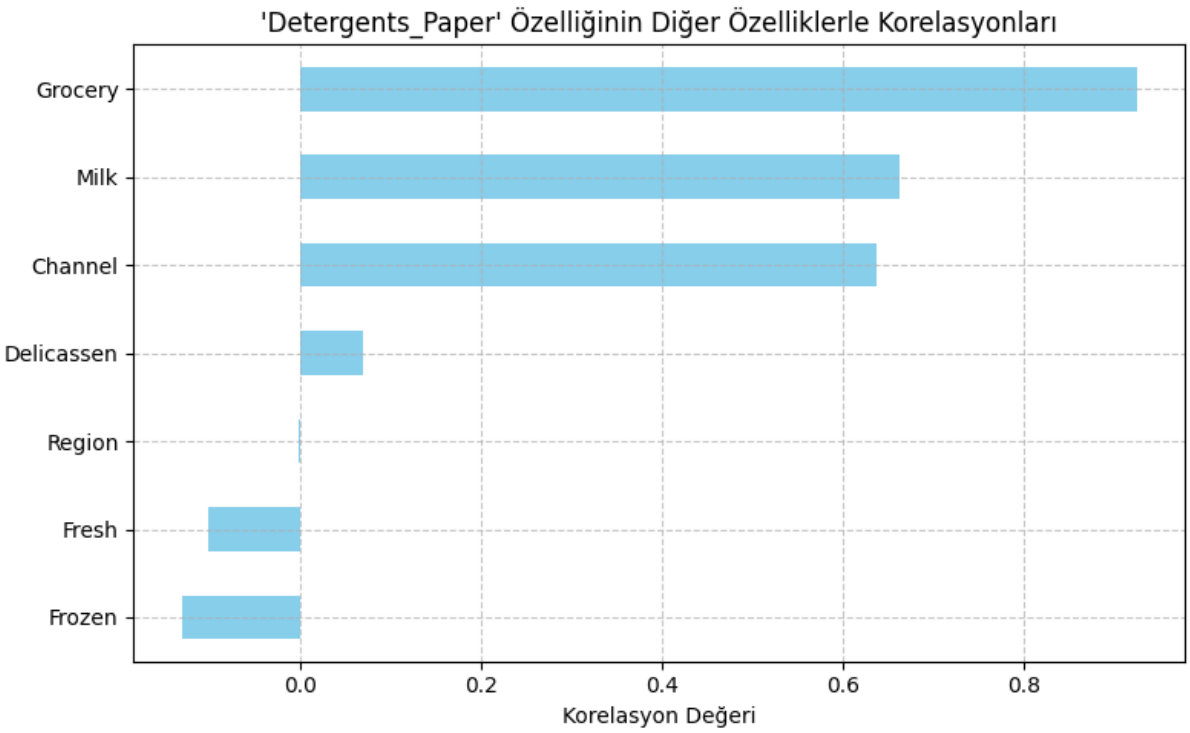
```
[ ]  import pandas as pd

     feature_importances = pd.Series(model.feature_importances_, index=X.columns)
     feature_importances = feature_importances.sort_values(ascending=False)

     print("Özellik Önemleri:")
     print(feature_importances)
```

```
Özellik Önemleri:
Detergents_Paper    0.919718
Milk                0.048932
Grocery             0.018027
Delicassen          0.013324
Region              0.000000
Fresh               0.000000
Frozen              0.000000
dtype: float64
```

4.7 Öznitelik Arasındaki Korelasyon İlişkilerinin Görselleştirilmesi

Veri setindeki öznitelikler arasındaki ilişkiyi daha iyi anlayabilmek için "Detergents_Paper" adlı özelliğin diğer değişkenlerle olan korelasyon değerlerini çubuk grafikle görselleştirdim. Bu sayede, hangi değişkenlerin birbiriyle daha güçlü ilişkiler kurduğunu kolayca gözlemleyebildim. Özellikle süt, bakkaliye ve şarküteri gibi bazı öznitelikler bu ürün grubuyla benzer harcama eğilimleri göstermiştir.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Korelasyon matrisini hesapla
corr_matrix = df.corr()

# Örneğin 'Detergents_Paper' ile diğer özniteliklerin korelasyonunu görselleştirelim
feature = 'Detergents_Paper'
correlations = corr_matrix[feature].drop(feature)  # Kendisiyle olan korelasyonu çıkar

# Çubuk grafiği çiz
plt.figure(figsize=(8,5))
correlations.sort_values().plot(kind='barh', color='skyblue')
plt.title(f"'{feature}' Özelliğinin Diğer Özelliklerle Korelasyonları")
plt.xlabel("Korelasyon Değeri")
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



5. Model Performans Değerlendirmesi

Aşağıdaki kod bloğu, test verisi üzerinde eğitilen karar ağacı modelinin doğruluk, kesinlik, geri çağırma, F1 skoru ve ROC AUC gibi sınıflandırma metriklerini hesaplamaktadır. Ayrıca modelin doğru ve yanlış sınıflandırmalarını gösteren karışıklık matrisi (confusion matrix) de çıkarılmıştır.

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix

# Tahminleri yap
y_pred = model.predict(X_test)

# Doğruluk (Accuracy)
accuracy = accuracy_score(y_test, y_pred)

# F1-Score
f1 = f1_score(y_test, y_pred)

# Recall
recall = recall_score(y_test, y_pred)

# Precision
precision = precision_score(y_test, y_pred)

# AUC Skoru (probabilistic output gerekli)
y_proba = model.predict_proba(X_test)[:, 1]
auc = roc_auc_score(y_test, y_proba)

# Confusion Matrix (TP, TN, FP, FN görmek için)
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()

# Sonuçları yazdır
print("Doğruluk (Accuracy):", round(accuracy, 4))
print("F1-Score:", round(f1, 4))
print("Recall:", round(recall, 4))
print("Precision:", round(precision, 4))
print("AUC:", round(auc, 4))
print("Confusion Matrix: TN =", tn, ", FP =", fp, ", FN =", fn, ", TP =", tp)
```

```
Doğruluk (Accuracy): 0.8864
F1-Score: 0.9206
Recall: 0.8923
Precision: 0.9508
AUC: 0.9261
Confusion Matrix: TN = 58 , FP = 7 , FN = 3 , TP = 20
```

```
[ ]  from sklearn.metrics import accuracy_score

     # Eğitim verisiyle tahmin yap
     y_train_pred = model.predict(X_train)

     # Eğitim doğruluk skorunu hesapla
     train_accuracy = accuracy_score(y_train, y_train_pred)
     print("Train Accuracy:", round(train_accuracy, 4))
```

```
Train Accuracy: 0.9801
```

6. Makale Karşılaştırması

Benim çalışmamda kullanılan sınıflandırma algoritması **Decision Tree**'dir. Modelin test setindeki başarımına göre:

- **Doğruluk (Accuracy)**: 88.64%
- **F1-Score**: 0.9206
- **Recall**: 0.8923
- **Precision**: 0.9508
- **AUC**: 92.61
- **Train Accuracy**: 98.01%

Kaggle'da Patel (n.d.) tarafından yayınlanan çalışmada ise farklı algoritmalar denenmiştir (KNN, SVM, Naive Bayes). Bu algoritmalardan en başarılısı test doğruluğu açısından KNN (90.91%) gibi görünse de, F1-Score, AUC, ve Precision gibi dengeli sınıflandırma metriklerinde benim kullandığım Decision Tree modeli daha iyi sonuç vermiştir.

Sonuç olarak, KNN daha yüksek bir test doğruluğu sağlasa da, Decision Tree daha dengeli bir performans ortaya koymuş ve sınıflar arasında daha tutarlı bir ayrım yapmıştır. Özellikle F1-Score ve Precision değerleri açısından öne çıkmaktadır.

Model	Train Accuracy	Test Accuracy	F1-Score	Recall	Precision	AUC	Best Cross Val Score	FN	FP	TN	TP
KNN	92.61%	90.91%	0.78	0.82	0.74	87.66	0.93 (+/- 0.02)	3	5	66	14
SVM	93.18%	89.77%	0.73	0.71	0.75	82.48	0.91 (+/- 0.00)	5	4	67	12
Naive Bayes	92.33%	88.64%	0.72	0.76	0.68	84.01	0.91 (+/- 0.00)	4	6	65	13
Decision Tree	98.01%	88.64%	0.9206	0.8923	0.9508	92.61	-	3	7	58	20

7. Kaynakça

- Patel, S. (n.d.). *Wholesale customer segmentation* [Notebook]. Kaggle. https://www.kaggle.com/code/sahistapatel96/wholesale-customer-segmentation
- OpenAI. (2025). *Google Colab notebook* [Online notebook]. https://colab.research.google.com/drive/1oLABzz9YQysLSEdDHb3XGs5AYCMRLiby?authuser=1#scrollTo=Qw_UxP2QQJxy
- Python Software Foundation. (n.d.). *Downloads* [Web sayfası]. https://www.python.org/downloads/
- Dua, D., & Graff, C. (2019). *UCI Machine Learning Repository: Wholesale customers data set*. University of California, Irvine, School of Information and Computer Sciences. https://archive.ics.uci.edu/dataset/292/wholesale+customers
- The Pandas Development Team. (n.d.). *pandas documentation*. https://pandas.pydata.org/pandas-docs/stable/
- Scikit-learn developers. (n.d.). *scikit-learn: Machine learning in Python*. https://scikit-learn.org/stable/
- Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering, 9(3), 90-95. https://matplotlib.org/
- Waskom, M. (2021). *Seaborn: Statistical data visualization*. Journal of Open Source Software, 6(60), 3021. https://seaborn.pydata.org/
- Harris, C. R., et al. (2020). *Array programming with NumPy*. Nature, 585(7825), 357-362. https://numpy.org/

8. Sunum Videosu, Dataset, Github Repo

- https://youtu.be/r4IkMQvrdE
- https://archive.ics.uci.edu/dataset/292/wholesale+customers
- https://github.com/sidikafirat/Wholesale-Classification