BURSA TEKNİK ÜNİVERSİTESİ

Mühendislik ve Doğa Bilimleri Fakültesi – Bilgisayar Mühendisliği Bölümü

BLM0463 Veri Madenciliğine Giriş

2023-2024 Bahar Dönemi

Final Ödevi Raporu

Berkan SERBES – 22360859353

İçindekiler

1.Giriş

Bu rapor, **Online Retail** veri seti kullanılarak yapılan sınıflandırma çalışmasının detaylı bir değerlendirmesini sunmaktadır. Modelin performansı, veri ön işleme adımları ve sonuçlar ayrıntılı bir şekilde ele alınmıştır. Bu sayede, çalışmanın metodolojisi ve elde edilen bulgular hakkında kapsamlı bir bilgi sağlanmıştır.

1.1 Problem Tanımı

Bu çalışma, UCI Machine Learning Repository'den alınan Online Retail veri seti kullanılarak müşteri davranışlarını sınıflandırmayı amaçlamaktadır. Veri seti, İngiltere merkezli bir çevrimiçi perakendeciden alınan işlemleri içermekte ve alışveriş miktarlarını pozitif (satın alma gerçekleşmiş) ve negatif (iade edilmiş) olarak etiketlemeyi hedeflemektedir. Bu etiketleme, perakendecinin müşteri davranışlarını daha iyi anlamasına ve gelecekteki stratejilerini şekillendirmesine yardımcı olabilir.

1.2 Veri Seti

Online Retail veri seti, çeşitli özelliklere sahip binlerce işlem kaydını içermektedir. Bu özellikler arasında fatura tarihi, fatura numarası, müşteri kimliği, ülke, ürün miktarı ve birim fiyatı gibi bilgiler bulunmaktadır. Veri setinin analiz edilmesi, müşteri segmentasyonunu ve müşteri davranışlarının modellenmesini mümkün kılmaktadır. Veri setindeki işlemler, 2010-2011 yılları arasında gerçekleşmiştir ve özellikle perakende sektöründe müşteri iade davranışlarını anlamak için değerli bir kaynaktır.

1.3 Amaç

Bu çalışmada, **Decision Tree** algoritması kullanılarak müşterilerin alışveriş miktarlarının pozitif veya negatif olarak sınıflandırılması hedeflenmiştir. Bu sınıflandırma, perakendecinin hangi müşterilerin alışveriş yapma eğiliminde olduğunu, hangilerinin ise iade yapma eğiliminde olduğunu belirlemesine yardımcı olacaktır. Böylece, müşteri memnuniyetini artıracak stratejiler geliştirilmesi mümkün olacaktır. Ayrıca, bu çalışma ile veri ön işleme adımları, model eğitimi ve performans değerlendirme süreçleri detaylı bir şekilde ele alınacaktır.

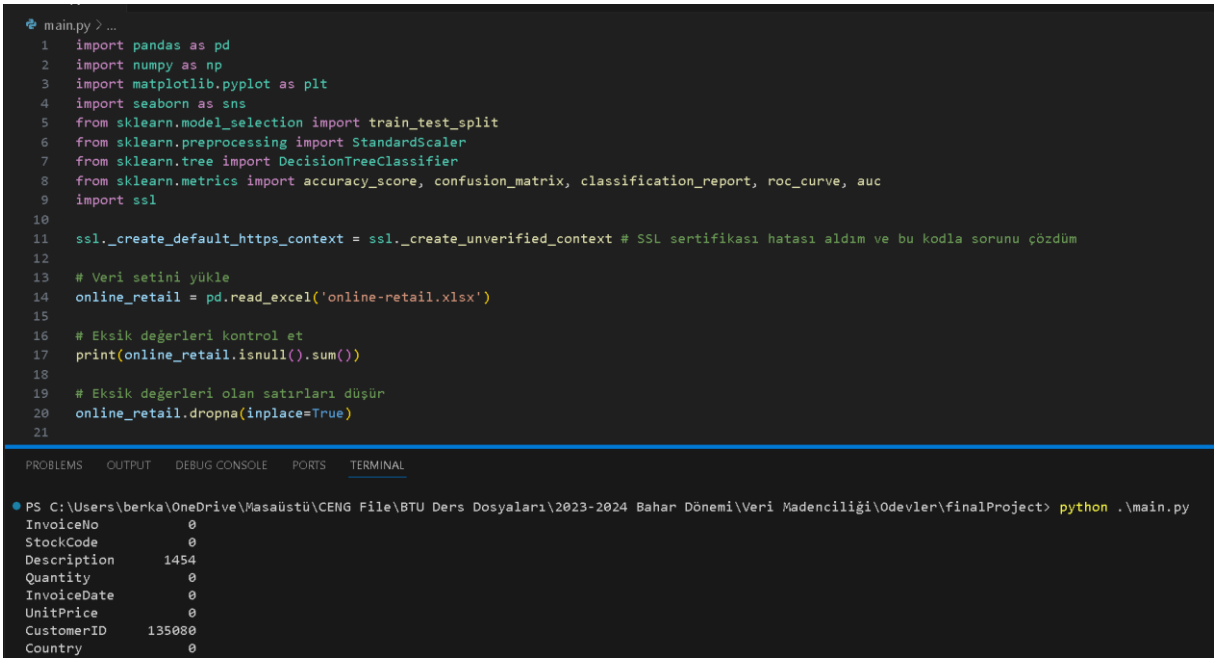
2. Gelişme

2.1 Veri Ön İşleme

Veri ön işleme, makine öğrenmesi projelerinde kritik bir adımdır. Veri setindeki eksik veya hatalı verilerin temizlenmesi, veri tiplerinin uygun formatlara dönüştürülmesi ve kategorik verilerin sayısal değerlere dönüştürülmesi bu adımın bir parçasıdır. Bu çalışmada, aşağıdaki veri ön işleme adımları uygulanmıştır:

1. **Eksik Verilerin Temizlenmesi:**
   Veri setindeki eksik değerler tespit edilmiş ve eksik değerlere sahip satırlar çıkarılmıştır. Eksik veri içeren satırlar, analiz sonuçlarının doğruluğunu olumsuz etkileyebileceği için bu adım önemlidir.

Aşağıdaki kod parçası, bir Excel dosyasından Online Retail veri setini yükler ve eksik verileri kontrol eder. İlk olarak, pandas kütüphanesi kullanılarak veri seti online_retail adlı bir DataFrame'e yüklenir. Ardından, **isnull().sum()** fonksiyonu ile her sütundaki eksik değerler sayılır ve ekrana yazdırılır. Son olarak, dropna(inplace=True) fonksiyonu ile eksik değer içeren satırlar veri setinden çıkarılır, böylece veri temizlenmiş ve analiz için hazır hale getirilmiş olur.



2. **Veri Tiplerinin Düzenlenmesi:**
   'InvoiceDate' kolonu datetime tipine dönüştürülmüş ve bu kolondan ay, gün ve saat bilgileri çıkarılmıştır. Bu, işlemlerin zamansal analizi için gereklidir.

```
# InvoiceDate kolonunu datetime tipine dönüştür
online_retail['InvoiceDate'] = pd.to_datetime(online_retail['InvoiceDate'])

# Ay, gün ve saat kolonlarını oluştur
online_retail['Month'] = online_retail['InvoiceDate'].dt.month
online_retail['Day'] = online_retail['InvoiceDate'].dt.day
online_retail['Hour'] = online_retail['InvoiceDate'].dt.hour
```

3. **Hedef Değişkenin Oluşturulması:**
   Quantity kolonu kullanılarak pozitif ve negatif etiketli bir hedef değişken (QuantityLabel) oluşturulmuştur. Pozitif miktarlar 1, negatif miktarlar ise 0 olarak etiketlenmiştir. Bu etiketleme, sınıflandırma modeli için hedef değişkeni belirlemede kullanılmıştır.

```
# Quantity kolonunu hedef değişken olarak kullanarak pozitif ve negatif olarak etiketle
online_retail['QuantityLabel'] = np.where(online_retail['Quantity'] > 0, 1, 0)
```

4. **Kategorik Değişkenlerin Dönüştürülmesi:**
   'Country' kolonu **one-hot encoding** yöntemiyle sayısal verilere dönüştürülmüştür. Kategorik değişkenlerin sayısal verilere dönüştürülmesi, modelin bu değişkenleri işleyebilmesi için gereklidir.

```
# Kategorik kolonları one-hot encoding ile dönüştür
online_retail = pd.get_dummies(online_retail, columns=['Country'])

# İlk birkaç satırı tekrar görüntüle
print(online_retail.head())
```

2.2 Özelliklerin (Attribute) ve Hedef (Target) Değişkenin Ayrılması

Özellikler (features) ve hedef değişken (target) şu şekilde ayrılmıştır:

- Özellikler: **'UnitPrice'**, **'Month'**, **'Day'**, **'Hour'** ve **'Country'** kolonları.
- Hedef Değişken: **'QuantityLabel'**

```
# Özellikler ve hedef değişkeni ayır
X = online_retail[['UnitPrice', 'Month', 'Day', 'Hour'] + [col for col in online_retail.columns if 'Country_' in col]]
y = online_retail['QuantityLabel']
```

2.3 Verinin Eğitim ve Test Setlerine Bölünmesi

Veri, eğitim (%70) ve test (%30) setlerine bölünmüştür. Bu sayede modelin performansı bağımsız bir test seti üzerinde değerlendirilebilmiştir. Eğitim seti modelin öğrenmesi için kullanılırken, test seti modelin genelleme yeteneğini değerlendirmek için kullanılmıştır.

```
# Veriyi eğitim ve test setlerine böl
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Veriyi standardize et
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

2.4 Modelin Eğitilmesi

Decision Tree algoritması kullanılarak model eğitilmiştir. Model eğitimi sırasında standartlaştırma (standardization) işlemi uygulanmış ve veriler aynı ölçeğe getirilmiştir. Standartlaştırma, özelliklerin ortalama etrafında toplanmasını ve varyanslarının azaltılmasını sağlar, böylece modelin daha iyi performans göstermesi beklenir.

```
# Decision Tree modelini oluştur ve eğit
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train_scaled, y_train)

# Test seti üzerinde tahminler yap
y_pred = dt_model.predict(X_test_scaled)
```

2.5 Model Performansının Değerlendirilmesi

Modelin performansı, test seti kullanılarak çeşitli metriklerle değerlendirilmiştir:

- Doğruluk (Accuracy): Doğruluk, modelin tüm sınıflandırma örnekleri üzerinde doğru tahminlerde bulunma oranını ifade eder. Bu, doğru tahminlerin toplam tahminlere oranı olarak hesaplanır.
  Doğruluk = (TP + TN) / N

- Hassasiyet (Sensitivity): Hassasiyet, modelin pozitif sınıfları doğru bir şekilde tahmin etme yeteneğini ölçer. Yani, gerçekte pozitif olan örneklerin ne kadarının doğru bir şekilde pozitif olarak tahmin edildiğini gösterir.
  Hassasiyet = TP / (TP / FN)

- Özgüllük (Specifity): Özgüllük, modelin negatif sınıfları doğru bir şekilde tahmin etme yeteneğini ölçer. Bu, gerçekte negatif olan örneklerin ne kadarının doğru bir şekilde negatif olarak tahmin edildiğini gösterir.
  Özgüllük = TN / (TN + FP)

- F-Measure: F-Measure (F1 Skoru olarak da bilinir), hassasiyet ve özgüllük arasında bir denge kurarak modelin genel performansını değerlendiren bir metriktir. F-Measure, özellikle sınıflar arasındaki dengesizlik durumlarında, modelin performansını daha bütünsel bir şekilde değerlendirir. Hassasiyet ve özgüllüğün harmonik ortalaması olarak hesaplanır ve her iki metriğin de dikkate alındığı bir denge sağlar.
  F-Measure = 2 x ((Hassasiyet x Özgüllük) / (Hassasiyet + Özgüllük))

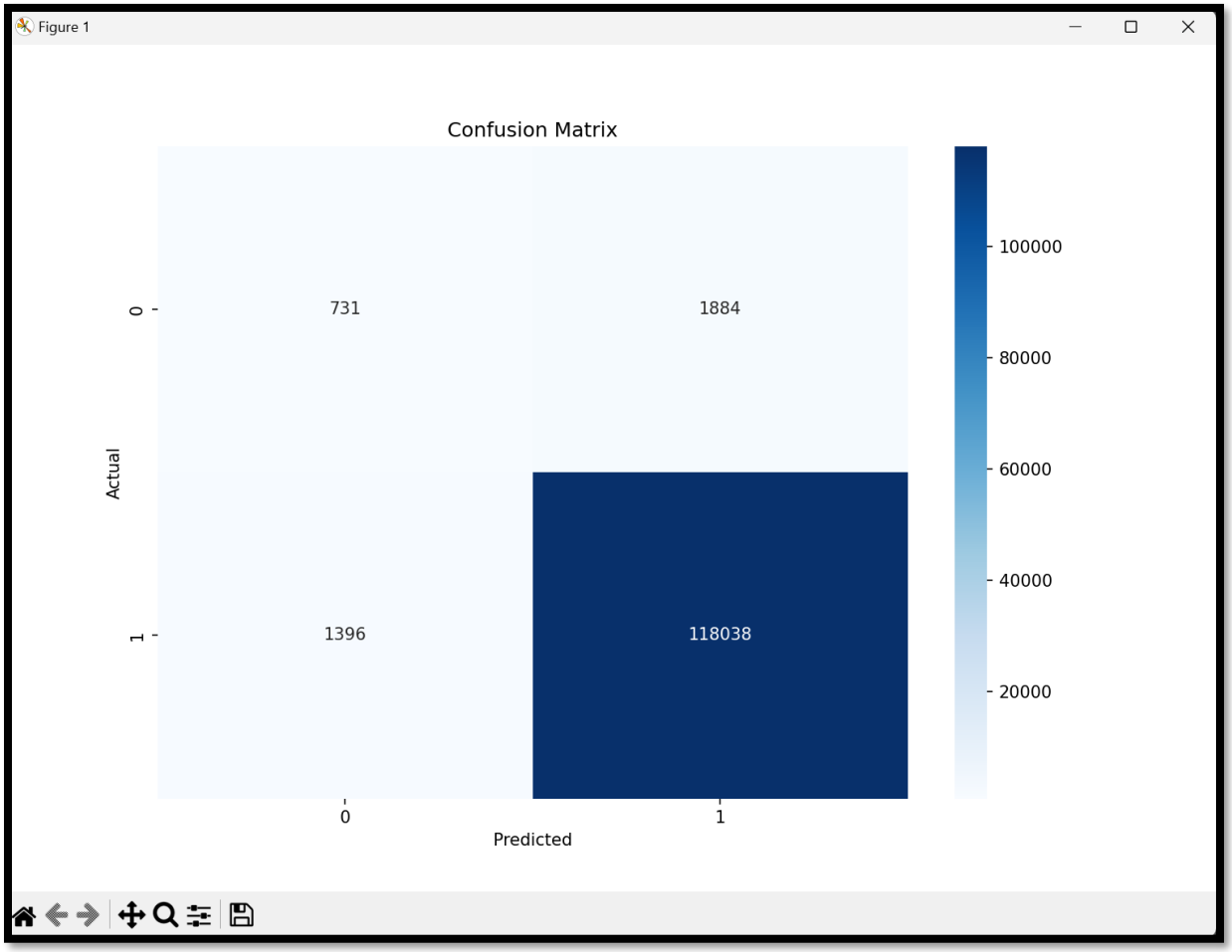
```
57
58  # Performans metriklerini hesapla
59  accuracy = accuracy_score(y_test, y_pred)
60  sensitivity = recall_score(y_test, y_pred, pos_label=1)
61  recall = recall_score(y_test, y_pred)
62  specificity = recall_score(y_test, y_pred, pos_label=0)
63  precision = precision_score(y_test, y_pred)
64  f1_score = f1_score(y_test, y_pred)
65  conf_matrix = confusion_matrix(y_test, y_pred)
66  class_report = classification_report(y_test, y_pred)
67
68
69  # Sonuçları yazdır
70  print(f'Accuracy: {accuracy}')
71  print(f'Sensitivity: {sensitivity}')
72  print(f'Recall: {recall}')
73  print(f'Precision: {precision}')
74  print(f'Specificity: {specificity}')
75  print(f'F1 Score: {f1_score}')
76  print('Confusion Matrix:')
77  print(conf_matrix)
78  print('Classification Report:')
79  print(classification_report(y_test, y_pred))
80
```

2.6 Model Performansının Sonuçları

Modelin test seti üzerindeki performans metrikleri şu şekildedir:

```
Accuracy: 0.9731255479356652
Sensitivity: 0.9883115360785036
Recall: 0.9883115360785036
Precision: 0.9842897883624356
Specificity: 0.27954110898661566
F1 Score: 0.9862965624425541
```

Confusion Matrix, modelin pozitif ve negatif sınıfları doğru ve yanlış sınıflandırma sayısını görselleştirmiştir. Aşağıda Confusion Matrix'in görselleştirilmiş hali bulunmaktadır:



Kodları ise aşağıdaki şekildeki gibidir.

```
# Confusion Matrix'i görselleştir
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

ROC (Receiver Operating Characteristic) eğrisi, bir sınıflandırma modelinin performansını değerlendirmenin yaygın bir yoludur. ROC eğrisi, modelin çeşitli eşik değerlerinde doğru pozitif oranı (True Positive Rate - TPR) ve yanlış pozitif oranı (False Positive Rate - FPR) arasındaki ilişkiyi gösterir. FPR x eksenini, TPR ise y eksenini ifade eder.

ROC Eğrisi Bileşenleri:

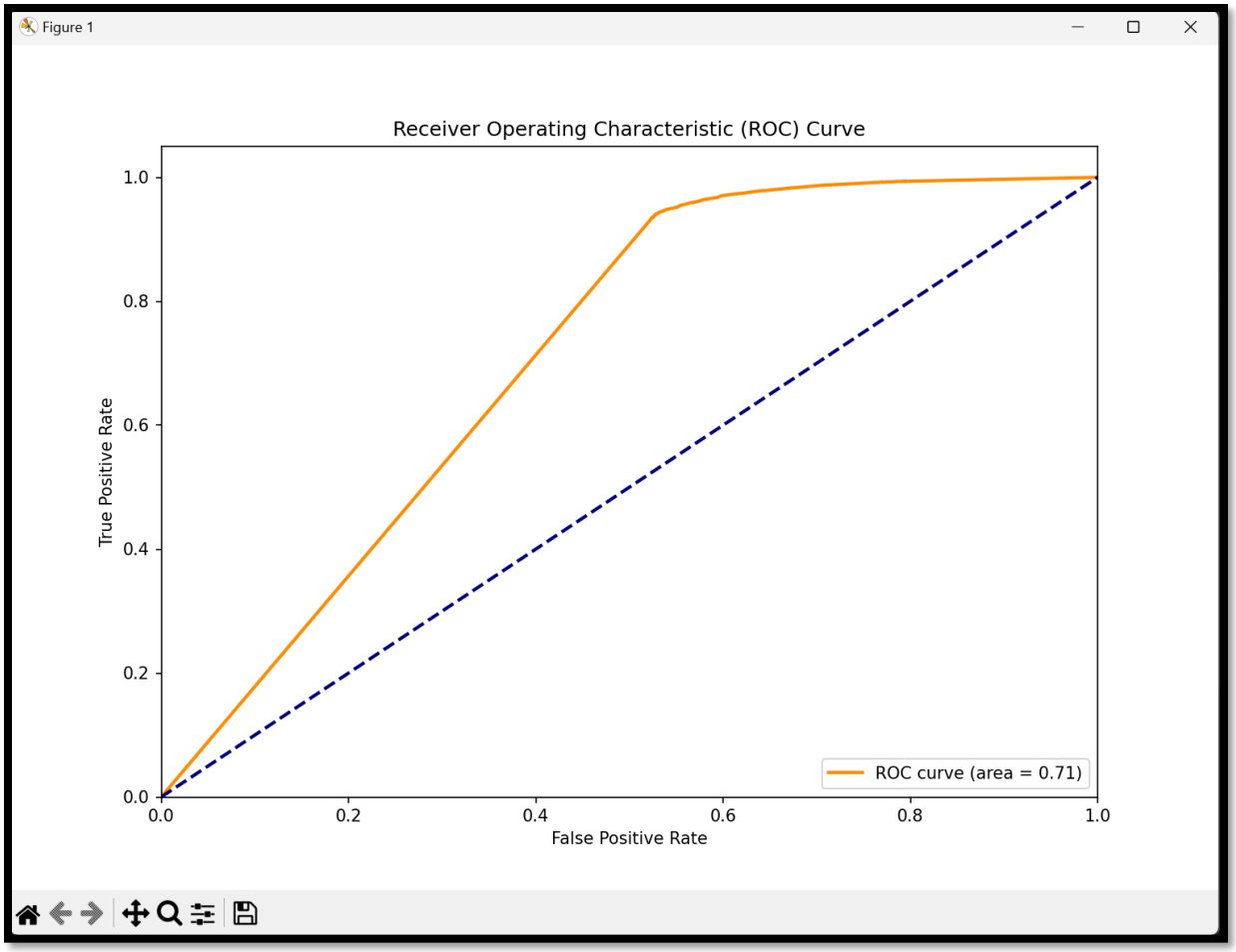
- **True Positive Rate (TPR) / Sensitivity (Duyarlılık):** Gerçek pozitiflerin, toplam gerçek pozitiflere oranıdır. Yani, doğru olarak pozitif olarak sınıflandırılan örneklerin, toplam pozitif örnekler içindeki oranıdır.
  TPR = FP / (FP + TN)
- **False Positive Rate (FPR):** Yanlış pozitiflerin, toplam gerçek negatiflere oranıdır. Yani, yanlış olarak pozitif olarak sınıflandırılan örneklerin, toplam negatif örnekler içindeki oranıdır.
  FPR = FP / (FP + TN)

AUC (Area Under the Curve): ROC eğrisinin altındaki alanı temsil eder ve modelin genel performansını tek bir değerle özetler. AUC değeri 0.5 ile 1 arasında değişir; 1 mükemmel bir sınıflandırmayı, 0.5 ise rastgele bir sınıflandırmayı temsil eder.

ROC eğrisini hesaplayan ve görselleştiren kod bloğu aşağıdaki gibidir.

```
# ROC eğrisini hesapla ve görselleştir
y_pred_proba = dt_model.predict_proba(X_test_scaled)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(10, 7))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```



3.Akademik Çalışma Karşılaştırması

Akademik Çalışma: https://arno.uvt.nl/show.cgi?fid=160793

Referans aldığımız çalışma, Tilburg Üniversitesi'nde yürütülen "Predicting Customer Return in Online Retail" başlıklı yüksek lisans tezidir. Bu tezde, aynı veri seti kullanılarak müşteri davranışları ve geri dönüş oranları tahmin edilmiştir. Tezde kullanılan yöntemler arasında Logistic Regression, Decision Trees ve Random Forest gibi çeşitli makine öğrenimi algoritmaları bulunmaktadır.

Çalışmanın Bulguları:

Logistic Regression: Bu model, %83.2 doğruluk oranı ile en iyi performansı göstermiştir.
Decision Trees: Karar ağaçları modeli, **%79.6** doğruluk oranı (accuracy) ile iyi bir performans sergilemiştir.
Random Forest: %85.3 doğruluk oranı ile en yüksek performansı göstermiştir

Kendi sonuçlarımız ile karşılaştırdığımızda, referans çalışmada kullanılan Random Forest modelinin daha düşük doğruluk oranına sahip olduğu görülmektedir.

4. Sonuç

Bu çalışma, Online Retail veri seti kullanılarak müşterilerin alışveriş miktarlarını pozitif ve negatif olarak sınıflandırmayı hedeflemiştir. Decision Tree algoritması kullanılarak eğitilen model, test seti üzerinde yüksek bir doğruluk oranı ile başarılı sonuçlar vermiştir. Modelin performans metrikleri, Decision Tree algoritmasının bu tür sınıflandırma problemleri için uygun olduğunu göstermektedir.

KAYNAKÇA

https://arno.uvt.nl/show.cgi?fid=160793

https://chatgpt.com/