# **PORJE 1 : DUYGU ANALİZİ (SENTİMENTANALYSİS)**

**RAPOR HAZIRLAYAN**

**FURKAN DEMİR**

df1["sentiment"] = df1["rating"].apply(lambda x: "pozitif" if x >= 3 else "negatif")

df2["sentiment"] = df2["rating"].apply(lambda x: "pozitif" if x >= 3 else "negatif")

df3["sentiment"] = df3["rating"].apply(lambda x: "pozitif" if x >= 3 else "negatif")

df = pd.concat([df1, df2, df3], axis=0)

**1. Sentiment Sütununu Ekleme:**

Kod, her bir DataFrame (df1, df2, df3) için aşağıdaki işlemi yapar:

* df1["sentiment"] = df1["rating"].apply(lambda x: "pozitif" if x >= 3 else "negatif")
* df2["sentiment"] = df2["rating"].apply(lambda x: "pozitif" if x >= 3 else "negatif")
* df3["sentiment"] = df3["rating"].apply(lambda x: "pozitif" if x >= 3 else "negatif")

Bu satırlar:

* df1, df2 ve df3 DataFrame'lerinde bulunan her bir "rating" değeri üzerinden apply fonksiyonu ile işlem yapar.
* Eğer rating 3 veya daha büyükse, "sentiment" sütunu "pozitif" olarak atanır.
* Eğer rating 3'ten küçükse, "sentiment" sütunu "negatif" olarak atanır.

**2. DataFrame'lerin Birleştirilmesi:**

* df = pd.concat([df1, df2, df3], axis=0) satırı ile üç DataFrame birleştirilir. Bu işlem:
  + axis=0 parametresi ile DataFrame'leri satır bazında (yani dikey olarak) birleştirir.
  + Bu işlem, üç DataFrame'in satırlarını birleştirir, yani her birinin üst üste eklenmesini sağlar.

**3. Sonuç:**

Birleştirilmiş DataFrame (df) şu şekilde olacak:

* Her üç DataFrame'in verileri birleştirilmiş olacak.
* Her DataFrame'deki "rating" değeri doğrultusunda "sentiment" sütunu oluşturulmuş olacak.

**Adımlar:**

1. rating değerine dayalı olarak sentiment sütunu oluşturuldu.
2. Üç DataFrame birleştirildi ve sonuç olarak tüm veriler tek bir DataFrame'de toplandı.

**Analiz:**

* Yeni eklenen sentiment sütunu, verileri pozitif ve negatif olarak iki grupta sınıflandırır.
* DataFrame'lerdeki "rating" değerlerinin dağılımı, bu sınıflandırmaların ne kadar yaygın olduğunu gösterir. Bu tür bir analiz, sentiment analizine dayalı kararlar almak için faydalı olabilir.
* Birleştirilen DataFrame, df isimli yeni bir DataFrame olarak tüm verileri içerir.

**Olası Kullanımlar:**

* Pozitif ve negatif yorumların oranını görmek için sentiment sütununu gruplandırıp sayabilirsiniz.
* Veri seti büyüdükçe, bu analiz ile kullanıcıların rating değerlerine göre eğilimlerini daha iyi anlayabilirsiniz.

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

train\_df, test\_df = train\_test\_split(df, test\_size=0.2, random\_state=42, shuffle=True)

**1. Veri Setinin Bölünmesi:**

Kodda, train\_test\_split fonksiyonu kullanılarak df veri seti eğitim ve test verisi olarak ikiye ayrılır:

* **Eğitim Verisi (train\_df)**: Bu, modelin öğrenme sürecinde kullanılacak veridir. Veri setinin %80'i eğitim verisi olarak ayrılır.
* **Test Verisi (test\_df)**: Modelin performansını değerlendirmek için kullanılacak veridir. Veri setinin %20'si test verisi olarak ayrılır.

**2. Parametreler:**

* **test\_size=0.2**: Veri setinin %20'si test verisi olarak ayrılır (geri kalan %80 eğitim için kullanılır).
* **random\_state=42**: Veri bölme işleminin tekrar edilebilir olması için rastgelelikin sabitlenmesini sağlar. Bu sayede her seferinde aynı bölme yapılır.
* **shuffle=True**: Veri rastgele karıştırılır. Bu, verinin sıralama düzeninin modelin eğitimini etkilememesi için önemlidir.

**3. Sonuç:**

* **train\_df**: Eğitim verilerini içerir.
* **test\_df**: Test verilerini içerir.

**Özet:**

* **Eğitim ve test verilerine bölünme** işlemi gerçekleştirmiş durumdayım , böylece modelin eğitim ve değerlendirilmesi için uygun veri setleri oluşturulmuştur.

# Gerekli kütüphaneleri içe aktarma

import re

import string

import nltk

from nltk.corpus import stopwords

from sklearn.feature\_extraction.text import TfidfVectorizer

# NLTK stopwords'ü indir (Eğer zaten yüklü değilse)

nltk.download('stopwords')

# İngilizce durak kelimeleri (stop words) al

stop\_words = set(stopwords.words('english'))

# Metin temizleme fonksiyonu

def clean\_text(text):

if isinstance(text, str):

text = text.lower() # Küçük harfe çevir

text = re.sub(f"[{string.punctuation}]", "", text) # Noktalama işaretlerini kaldır

text = " ".join([word for word in text.split() if word not in stop\_words]) # Stopwords kaldır

return text

return ""

# Train ve Test setlerindeki metinleri temizleme

train\_df["clean\_text"] = train\_df["text"].apply(clean\_text)

test\_df["clean\_text"] = test\_df["text"].apply(clean\_text)

# TF-IDF vektörleştirme

tfidf\_vectorizer = TfidfVectorizer(max\_features=5000) # 5000 kelimeye kadar vektörize et

X\_train\_tfidf = tfidf\_vectorizer.fit\_transform(train\_df["clean\_text"])

X\_test\_tfidf = tfidf\_vectorizer.transform(test\_df["clean\_text"])

Metin verisini temizleyip, TF-IDF (Term Frequency - Inverse Document Frequency) yöntemiyle sayısal verilere dönüştürmek.

**Adımlar:**

1. **Kütüphane ve Veri Seti Hazırlığı:**
   * Gerekli kütüphaneler (nltk, re, string, TfidfVectorizer) içe aktarılmıştır.
   * **NLTK** kullanılarak İngilizce stopwords (durak kelimeler) indirilmiştir.
2. **Metin Temizleme Fonksiyonu:**
   * **clean\_text** fonksiyonu oluşturulmuştur:
     + Metni küçük harfe çevirir.
     + Noktalama işaretlerini kaldırır.
     + Stopwords (önemsiz kelimeler) listesine göre metni temizler.
3. **Metin Temizleme:**
   * **train\_df** ve **test\_df** veri setlerinde yer alan metinler, clean\_text fonksiyonu ile temizlenmiştir. Temizlenmiş metinler clean\_text adlı yeni bir sütunda saklanmıştır.
4. **TF-IDF Vektörleştirme:**
   * **TF-IDF** ile metin verisi sayısal verilere dönüştürülmüştür.
   * max\_features=5000 parametresi ile her bir metin için en fazla 5000 kelime kullanılacak şekilde vektörleştirme yapılmıştır.
   * **fit\_transform** eğitim verisinde, **transform** ise test verisinde uygulanmıştır.

**Sonuç:**

* Temizlenmiş metin verileri, sayısal verilere dönüştürülerek modelin eğitimi için hazır hale getirilmiştir.
* Eğitim ve test veri setlerinin temizlenmiş ve vektörleştirilmiş metinleri (X\_train\_tfidf ve X\_test\_tfidf) elde edilmiştir.

**Özet:**

Bu adımlar, doğal dil işleme (NLP) süreçlerinde metin verisini hazırlamak ve sayısal hale getirmek için kullanılan temel adımlardır.

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, f1\_score, confusion\_matrix, classification\_report

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

Makine öğrenmesi modelleri kullanarak sınıflandırma işlemi yapmak ve model performansını değerlendirmek.

**Adımlar:**

1. **Kütüphanelerin İçeri Aktarılması:**
   * **LogisticRegression** ve **RandomForestClassifier** sınıflandırma algoritmaları için içe aktarılmıştır.
   * **accuracy\_score, f1\_score, confusion\_matrix, classification\_report** gibi metrikler, model performansını değerlendirmek için kullanılacaktır.
   * **numpy, seaborn, ve matplotlib** kütüphaneleri, görselleştirme ve sayısal işlemler için kullanılacaktır.
2. **Modellerin Eğitimi:**
   * **Lojistik Regresyon (LogisticRegression)** ve **Rastgele Orman Sınıflandırıcısı (RandomForestClassifier)** ile sınıflandırma modelleri oluşturulacaktır.
   * Her iki model de eğitim verisi ile eğitilecektir.
3. **Model Performans Değerlendirmesi:**
   * Modellerin başarılarını değerlendirmek için:
     + **accuracy\_score**: Modelin doğru tahminlerinin oranını hesaplar.
     + **f1\_score**: Modelin genel başarısını daha detaylı ölçer (özellikle dengesiz veri setlerinde kullanışlı).
     + **confusion\_matrix**: Modelin gerçek ve tahmin edilen sınıfları arasındaki ilişkiyi gösteren bir matrisi oluşturur.
     + **classification\_report**: Modelin doğruluk, precision, recall ve F1 skoru gibi istatistikleri içeren bir rapor oluşturur.
4. **Görselleştirme:**
   * **seaborn** ve **matplotlib** kütüphaneleri kullanılarak **confusion matrix** görselleştirilir. Bu, modelin hangi sınıflarda hatalar yaptığını görsel olarak analiz etmeyi sağlar.

**Sonuç:**

* İki farklı modelin eğitimi ve performans değerlendirmesi yapılacaktır.
* Performans metrikleri ve görselleştirmelerle modellerin başarıları karşılaştırılabilir.

**Özet:**

Bu adımlar, iki farklı makine öğrenmesi modelinin eğitilmesini ve performanslarının değerlendirilmesini içerir. Sonuçlar, modelin doğruluğunu ve sınıflandırma başarısını ölçmek için kullanılacaktır.

y\_train = train\_df["sentiment"].map({"pozitif": 1, "negatif": 0})

y\_test = test\_df["sentiment"].map({"pozitif": 1, "negatif": 0})

Metin sınıflandırma probleminin hedef değişkenini (etiketlerini) sayısal değerlere dönüştürmek.

**Adımlar:**

1. **Etiket Dönüşümü:**
   * train\_df["sentiment"] ve test\_df["sentiment"] sütunlarındaki metinsel sınıflar ("pozitif" ve "negatif") sayısal değerlere dönüştürülür:
     + **"pozitif"** etiketini **1** ile, **"negatif"** etiketini ise **0** ile eşleştirir.
     + Bu işlem için **map()** fonksiyonu kullanılır.
2. **Sonuç:**
   * **y\_train**: Eğitim verisi için sayısal hedef değişkeni (0 ve 1).
   * **y\_test**: Test verisi için sayısal hedef değişkeni (0 ve 1).

**Özet:**

Bu işlem, sınıflandırma modelinde kullanılacak hedef değişkenleri sayısal değerlere dönüştürerek modelin doğru şekilde öğrenmesini sağlar. "Pozitif" ve "Negatif" etiketleri sırasıyla 1 ve 0 ile eşlenmiştir.

logreg = LogisticRegression(max\_iter=10)

logreg.fit(X\_train\_tfidf, y\_train)

y\_pred\_logreg = logreg.predict(X\_test\_tfidf)

Lojistik Regresyon modeli ile metin sınıflandırması yapmak ve test verisi üzerinde tahminlerde bulunmak.

**Adımlar:**

1. **Modelin Eğitilmesi:**
   * **Lojistik Regresyon (LogisticRegression)** modeli, eğitim verisi (X\_train\_tfidf) ve hedef etiketler (y\_train) kullanılarak eğitilir.
   * Modelin maksimum iterasyon sayısı **10** olarak belirlenmiştir (max\_iter=10).
2. **Tahmin Yapılması:**
   * Eğitim tamamlandıktan sonra, model test verisi (X\_test\_tfidf) üzerinde tahminler yapmak için kullanılır.
   * Tahminler **y\_pred\_logreg** değişkenine kaydedilir.

**Sonuç:**

* Lojistik Regresyon modeli eğitildikten sonra, test verisi üzerinde tahminler yapılmış ve **y\_pred\_logreg** değişkeninde saklanmıştır.

**Özet:**

Bu adımlar, Lojistik Regresyon modelinin eğitilmesini ve test verisi üzerinde sınıflandırma tahminleri yapılmasını içerir. Modelin performansını değerlendirmek için tahmin sonuçları kullanılacaktır

rf = RandomForestClassifier(n\_estimators=5, random\_state=42)

rf.fit(X\_train\_tfidf, y\_train)

y\_pred\_rf = rf.predict(X\_test\_tfidf)

Rastgele Orman (Random Forest) modeli ile metin sınıflandırması yapmak ve test verisi üzerinde tahminlerde bulunmak.

**Adımlar:**

1. **Modelin Eğitilmesi:**
   * **Rastgele Orman Sınıflandırıcısı (RandomForestClassifier)** modeli, eğitim verisi (X\_train\_tfidf) ve hedef etiketler (y\_train) kullanılarak eğitilir.
   * Modeldeki ağaç sayısı **5** olarak belirlenmiştir (n\_estimators=5).
   * **random\_state=42** parametresi, modelin her çalıştırıldığında aynı sonuçları vermesini sağlar.
2. **Tahmin Yapılması:**
   * Model eğitildikten sonra, test verisi (X\_test\_tfidf) üzerinde tahminler yapılır.
   * Tahminler **y\_pred\_rf** değişkenine kaydedilir.

**Sonuç:**

* Rastgele Orman modeli eğitildikten sonra, test verisi üzerinde tahminler yapılmış ve sonuçlar **y\_pred\_rf** değişkeninde saklanmıştır.

**Özet:**

Bu adımlar, Rastgele Orman modelinin eğitimini ve test verisi üzerinde tahmin yapmayı içerir. Modelin doğruluğu ve performansı, tahmin sonuçları ile değerlendirilebilir.

models = {

"Logistic Regression": y\_pred\_logreg,

"Random Forest": y\_pred\_rf

}

İki farklı sınıflandırma modelinin (Lojistik Regresyon ve Rastgele Orman) tahmin sonuçlarını bir arada tutmak.

**Adımlar:**

1. **Model Tahmin Sonuçlarının Saklanması:**
   * **models** adında bir sözlük (dictionary) oluşturulmuştur.
   * Bu sözlük, iki modelin tahmin sonuçlarını tutar:
     + **"Logistic Regression"**: Lojistik Regresyon modelinin tahmin sonuçları **y\_pred\_logreg**.
     + **"Random Forest"**: Rastgele Orman modelinin tahmin sonuçları **y\_pred\_rf**.

**Sonuç:**

* **models** sözlüğü, iki modelin tahmin sonuçlarını saklar ve her iki modelin sonuçlarına kolayca erişim sağlar.

**Özet:**

Bu adım, iki farklı modelin tahmin sonuçlarını bir arada saklamak için bir sözlük oluşturur. Bu, her iki modelin performansını karşılaştırmak ve analiz etmek için faydalıdır.

metrics\_results = {}

for model\_name, y\_pred in models.items():

accuracy = accuracy\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

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

metrics\_results[model\_name] = {"Accuracy": accuracy, "F1 Score": f1, "Confusion Matrix": confusion}

plt.figure(figsize=(5, 4))

sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues', xticklabels=["Negatif", "Pozitif"], yticklabels=["Negatif", "Pozitif"])

plt.xlabel("Tahmin Edilen")

plt.ylabel("Gerçek")

plt.title(f"{model\_name} - Karışıklık Matrisi")

plt.show()

İki modelin (Lojistik Regresyon ve Rastgele Orman) performansını değerlendirmek ve karışıklık matrislerini görselleştirmek.

**Adımlar:**

1. **Performans Metriklerinin Hesaplanması:**
   * **metrics\_results** adlı boş bir sözlük oluşturulmuştur.
   * Her modelin tahmin sonuçları için şu metrikler hesaplanır:
     + **accuracy\_score**: Modelin doğruluk oranı.
     + **f1\_score**: Modelin F1 skoru (dengeyi ölçer).
     + **confusion\_matrix**: Modelin tahminleri ile gerçek değerler arasındaki ilişkiyi gösteren karışıklık matrisi.
2. **Sonuçların Saklanması:**
   * Hesaplanan metrikler, **metrics\_results** sözlüğünde model adı ile ilişkilendirilir.
3. **Karışıklık Matrisinin Görselleştirilmesi:**
   * **sns.heatmap** fonksiyonu ile her modelin karışıklık matrisi görselleştirilir.
   * Her karışıklık matrisi, **"Negatif"** ve **"Pozitif"** sınıflarını gösteren ısı haritası ile sunulur.
   * Görsellerde her model için **"Tahmin Edilen"** ve **"Gerçek"** etiketleri bulunur.

**Sonuç:**

* Model performans metrikleri (doğruluk, F1 skoru ve karışıklık matrisi) **metrics\_results** sözlüğünde saklanmıştır.
* Her modelin karışıklık matrisi görselleştirilmiş ve ekranda gösterilmiştir.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Özet:**

Bu adımlar, her iki modelin performansını değerlendirir, karşılaştırır ve her model için karışıklık matrislerini görsel olarak sunar. Bu sayede modelin doğruluğu, F1 skoru ve sınıflandırma hataları hakkında derinlemesine bilgi edinilebilir.

import tensorflow as tf

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout

from sklearn.preprocessing import LabelEncoder

MAX\_NUM\_WORDS = 10000

MAX\_SEQUENCE\_LENGTH = 200

EMBEDDING\_DIM = 100

tokenizer = Tokenizer(num\_words=MAX\_NUM\_WORDS, oov\_token="<OOV>")

tokenizer.fit\_on\_texts(train\_df["clean\_text"])

X\_train\_seq = tokenizer.texts\_to\_sequences(train\_df["clean\_text"])

X\_test\_seq = tokenizer.texts\_to\_sequences(test\_df["clean\_text"])

X\_train\_pad = pad\_sequences(X\_train\_seq, maxlen=MAX\_SEQUENCE\_LENGTH, padding="post")

X\_test\_pad = pad\_sequences(X\_test\_seq, maxlen=MAX\_SEQUENCE\_LENGTH, padding="post")

label\_encoder = LabelEncoder()

y\_train\_enc = label\_encoder.fit\_transform(train\_df["sentiment"])

y\_test\_enc = label\_encoder.transform(test\_df["sentiment"])

model = Sequential([

Embedding(MAX\_NUM\_WORDS, EMBEDDING\_DIM, input\_length=MAX\_SEQUENCE\_LENGTH),

LSTM(128, return\_sequences=True),

Dropout(0.3),

LSTM(64),

Dropout(0.3),

Dense(32, activation="relu"),

Dense(1, activation="sigmoid")

])

model.compile(loss="binary\_crossentropy", optimizer="adam", metrics=["accuracy"])

history = model.fit(X\_train\_pad, y\_train\_enc, validation\_data=(X\_test\_pad, y\_test\_enc),

epochs=1, batch\_size=32, verbose=1)

y\_pred\_lstm = (model.predict(X\_test\_pad) > 0.5).astype("int32")

accuracy = accuracy\_score(y\_test\_enc, y\_pred\_lstm)

f1 = f1\_score(y\_test\_enc, y\_pred\_lstm)

confusion = confusion\_matrix(y\_test\_enc, y\_pred\_lstm)

import seaborn as sns

import matplotlib.pyplot as plt

plt.figure(figsize=(5, 4))

sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues', xticklabels=["Negatif", "Pozitif"], yticklabels=["Negatif", "Pozitif"])

plt.xlabel("Tahmin Edilen")

plt.ylabel("Gerçek")

plt.title("LSTM Modeli - Karışıklık Matrisi")

plt.show()

Metin verisini LSTM (Long Short-Term Memory) modeline uygulayarak metin sınıflandırması yapmak ve modelin performansını değerlendirmek.

**Adımlar:**

1. **Veri Hazırlığı:**
   * **Tokenizer** kullanılarak metin verisi sayısal verilere dönüştürülür.
     + fit\_on\_texts ile eğitim verisindeki metinler üzerinde kelime dizileri oluşturulur.
     + texts\_to\_sequences ile metinler sayılara dönüştürülür.
   * **Padding** işlemi ile metin uzunlukları sabitlenir. maxlen=200 parametresi ile her metin 200 kelimelik sabit uzunlukta olacak şekilde pad edilir.
   * **LabelEncoder** kullanılarak hedef etiketler ("pozitif" ve "negatif") sayısal değerlere dönüştürülür.
2. **LSTM Modelinin Oluşturulması:**
   * **Sequential** model yapısı oluşturulmuştur.
   * Modelin katmanları:
     + **Embedding Layer**: Kelimeleri sayısal vektörlere dönüştürür.
     + **LSTM Layers**: Sırasıyla 128 ve 64 nöronlu iki LSTM katmanı içerir.
     + **Dropout Layers**: Overfitting'i önlemek için her LSTM katmanının ardından dropout uygulanır (0.3 oranında).
     + **Dense Layer**: Sonuçları sigmoid fonksiyonu ile 0-1 arasında sınıflandırır (binary classification).
   * Modelin **binary\_crossentropy** kayıp fonksiyonu ile derlenmesi ve **adam** optimizer'ı ile optimize edilmesi sağlanmıştır.
3. **Model Eğitimi:**
   * Model, eğitim verisi (X\_train\_pad ve y\_train\_enc) ile eğitilir, doğrulama ise test verisi (X\_test\_pad ve y\_test\_enc) ile yapılır.
   * 1 epoch ve 32 batch size ile eğitim gerçekleştirilmiştir.
4. **Tahmin ve Performans Değerlendirmesi:**
   * Model, test verisi üzerinde tahminler yapar (y\_pred\_lstm).
   * Modelin performansı:
     + **Accuracy** ve **F1 Skoru** hesaplanır.
     + **Karışıklık Matrisi** görselleştirilir.
5. **Karışıklık Matrisinin Görselleştirilmesi:**
   * **seaborn** ve **matplotlib** kullanılarak karışıklık matrisi görselleştirilir. "Negatif" ve "Pozitif" sınıflarının tahmin sonuçları görsel olarak sunulur.

**Sonuç:**

* LSTM modelinin eğitilmesi, tahmin yapılması ve modelin performansının değerlendirilmesi yapılmıştır.
* Sonuçlar, **accuracy**, **F1 skoru**, ve **karışıklık matrisi** ile raporlanmıştır.

**Özet:**

Bu kod, metin sınıflandırması için LSTM modelini kullanarak metin verisini sayısal hale getirir, modeli eğitir ve test eder. Modelin doğruluğu ve performansı, F1 skoru ve karışıklık matrisi ile değerlendirilmiştir. Bu adımlar, metin sınıflandırması için bir derin öğrenme modelinin tamamlanması sürecini kapsamaktadır.

* Model başarıyla eğitilmiş ve doğrulama verisinde iyi bir performans sergilemiş (doğruluk %90.21).
* Eğitim sırasında kayıp (loss) değeri 0.4832 iken, doğrulama kaybı 0.2325 olmuş, bu da modelin genelleme yeteneğini gösteriyor.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

import matplotlib.pyplot as plt

from wordcloud import WordCloud

# Eğer "clean\_text" sütunu yoksa, temizleme işlemi yapalım

if "clean\_text" not in train\_df.columns:

train\_df["clean\_text"] = train\_df["text"].astype(str).str.lower()

# Pozitif ve negatif yorumlar için ayrı word cloud oluşturma

positive\_text = " ".join(train\_df[train\_df["sentiment"] == "pozitif"]["clean\_text"])

negative\_text = " ".join(train\_df[train\_df["sentiment"] == "negatif"]["clean\_text"])

# Word Cloud görselleştirme

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

# Pozitif yorumlar için Word Cloud

plt.subplot(1, 2, 1)

wordcloud\_positive = WordCloud(width=500, height=300, background\_color="white").generate(positive\_text)

plt.imshow(wordcloud\_positive, interpolation="bilinear")

plt.axis("off")

plt.title("Pozitif Yorumlar - Word Cloud")

# Negatif yorumlar için Word Cloud

plt.subplot(1, 2, 2)

wordcloud\_negative = WordCloud(width=500, height=300, background\_color="black", colormap="Reds").generate(negative\_text)

plt.imshow(wordcloud\_negative, interpolation="bilinear")

plt.axis("off")

plt.title("Negatif Yorumlar - Word Cloud")

# Görselleştirme

plt.show()

Pozitif ve negatif yorumlar için ayrı ayrı **Word Cloud** görselleştirmeleri oluşturmak.

**Adımlar:**

1. **Veri Hazırlığı:**
   * Eğer **clean\_text** sütunu mevcut değilse, **text** sütunundan küçük harfe dönüştürülerek **clean\_text** sütunu oluşturulur.
2. **Pozitif ve Negatif Yorumların Ayrılması:**
   * **Pozitif Yorumlar:** **train\_df** veri çerçevesinde "pozitif" etiketine sahip yorumlar birleştirilerek tek bir metin haline getirilir.
   * **Negatif Yorumlar:** "Negatif" etiketine sahip yorumlar da birleştirilerek ayrı bir metin oluşturulur.
3. **Word Cloud Görselleştirme:**
   * **Pozitif Yorumlar için Word Cloud:**
     + Beyaz arka planla ve varsayılan renk paletiyle pozitif yorumlardan bir Word Cloud oluşturulur.
     + Görselde kelimeler, daha fazla geçen kelimeler daha büyük ve belirgin olur.
   * **Negatif Yorumlar için Word Cloud:**
     + Siyah arka plan ve kırmızı renk paletiyle negatif yorumlardan bir Word Cloud oluşturulur.
4. **Görselleştirme:**
   * Her iki Word Cloud'ı yan yana gösteren bir görsel oluşturulur.
   * Görsellerde, her bir Word Cloud için başlıklar eklenir: "Pozitif Yorumlar - Word Cloud" ve "Negatif Yorumlar - Word Cloud".

**Sonuç:**

* **Pozitif ve negatif yorumlar** için ayrı ayrı **Word Cloud** görselleştirmeleri oluşturulmuş ve görselleştirilmiştir. Bu görselleştirmeler, her iki sınıfın en sık kullanılan kelimelerini ve kelimelerin görsel yoğunluklarını gösterir.

**Özet:**

Bu kod, pozitif ve negatif yorumları ayırarak her biri için **Word Cloud** oluşturur ve iki görselleştirmeyi yan yana sunar. Bu sayede, her iki sınıfın içerdiği en yaygın kelimeleri görsel olarak keşfetmek mümkündür.

metin, yazı tipi, ekran görüntüsü, tipografi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

from wordcloud import WordCloud

positive\_text = " ".join(train\_df[train\_df["sentiment"] == "pozitif"]["clean\_text"])

negative\_text = " ".join(train\_df[train\_df["sentiment"] == "negatif"]["clean\_text"])

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

plt.subplot(1, 2, 1)

wordcloud\_positive = WordCloud(width=500, height=300, background\_color="white").generate(positive\_text)

plt.imshow(wordcloud\_positivnterpolation="bilinear")

plt.axis("off")

plt.title("Pozitif Yorumlar - Word Cloud")

plt.subplot(1, 2, 2)

wordcloud\_negative = WordCloud(width=500, height=300, background\_color="black", colormap="Reds").generate(negative\_text)

plt.imshow(wordcloud\_negative, interpolation="bilinear")

plt.axis("off")

plt.title("Negatif Yorumlar - Word Cloud")

plt.show()

Pozitif ve negatif yorumlar için **Word Cloud** görselleştirmeleri oluşturmak ve her iki sınıfın en sık kullanılan kelimelerini görsel olarak sunmak.

**Adımlar:**

1. **Veri Hazırlığı:**
   * **Pozitif Yorumlar:** "Pozitif" sentiment etiketine sahip yorumlar birleştirilir ve tek bir metin haline getirilir (positive\_text).
   * **Negatif Yorumlar:** "Negatif" sentiment etiketine sahip yorumlar birleştirilir ve tek bir metin haline getirilir (negative\_text).
2. **Word Cloud Görselleştirme:**
   * **Pozitif Yorumlar için Word Cloud:**
     + Beyaz arka plan ile pozitif yorumların sıklıkla kullanılan kelimeleri bir Word Cloud üzerinde görselleştirilir.
     + Bu görselde daha fazla tekrar eden kelimeler daha büyük görünür.
   * **Negatif Yorumlar için Word Cloud:**
     + Siyah arka plan ve kırmızı renk paletiyle negatif yorumların sıklıkla kullanılan kelimeleri bir Word Cloud üzerinde görselleştirilir.
3. **Görselleştirme:**
   * İki Word Cloud görseli yan yana yerleştirilir.
   * Her iki görsel için başlıklar belirlenir:
     + **Pozitif Yorumlar - Word Cloud**
     + **Negatif Yorumlar - Word Cloud**
4. **Sonuçları Gösterme:**
   * **plt.show()** komutu ile her iki Word Cloud görselleştirmesi ekranda görüntülenir.

**Sonuç:**

* Pozitif ve negatif yorumlar için ayrı ayrı **Word Cloud** görselleştirmeleri oluşturulmuş ve görselleştirilmiştir.
* Görselleştirmeler, her iki sınıfın içerdiği en yaygın kelimeleri ve kelimelerin yoğunluğunu görsel olarak gösterir.

**Özet:**

Bu kod, **pozitif ve negatif yorumlar** için **Word Cloud** görselleştirmelerini oluşturur ve her iki sınıfın sık kullanılan kelimelerini görsel olarak keşfetmek için yan yana sunar.