

# STEAM KULLANICI DAVRANIŞI DUYGU ANALİZİ PROJESİ



Github: <https://github.com/22eda/python-steam-sentiment-analysis>

Local Web Sitesi: <http://192.168.1.103:5000>

<http://127.0.0.1:5000>

Render web link: <https://python-steam-sentiment-analysis-6ins.onrender.com>

🎮 Proje Konusu:

Steam platformundaki oyunlara yapılan kullanıcı yorumları üzerinden Python diliyle duygu analizi yapan bir web sitesi oluşturmak , oyuncu davranışlarının analiz edilmesi.

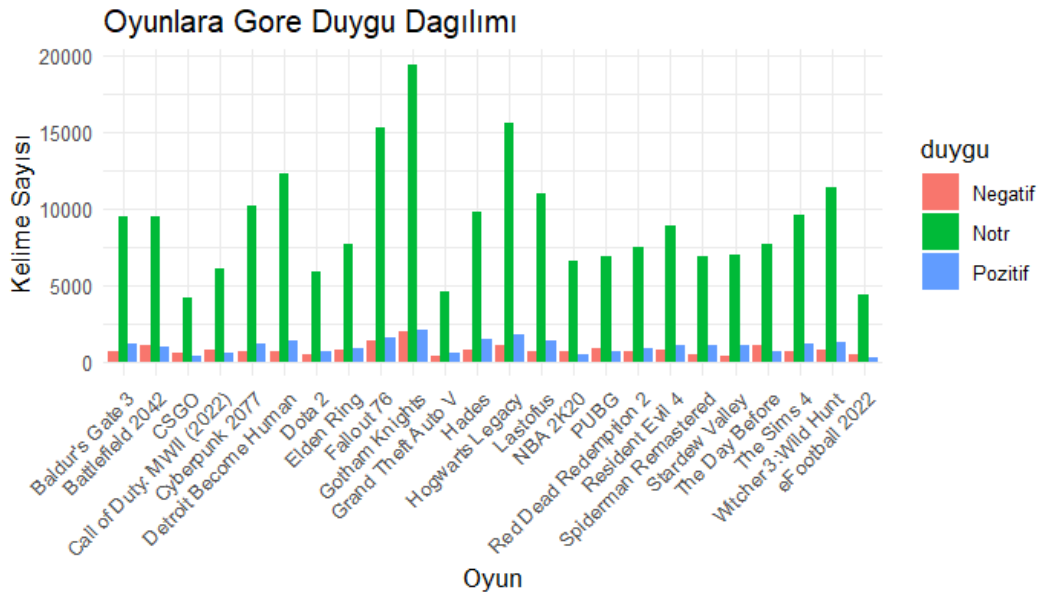
- Oyuncuların oyunlara dair bıraktığı yorumlar toplanıyor (pozitif/negatif/nötr).
- Bu yorumlar üzerinden **duygu analizi** (sentiment analysis) uygulanıyor.
- Sonuçlar oyunların popülerliği, oynanma süresi veya türüyle ilişkilendirilerek **kullanıcı davranışları** analiz ediliyor.

HAZIRLAYAN: EDANUR DEMİREL  
DERS: İST204

# Teknik Dokümantasyon ve Kod Analizi Raporu

## İÇİNDEKİLER

1. [Proje Genel Bakış](#)
2. [Sistem Mimarisi](#)
3. [Veri Toplama Süreci](#)
4. [Veri İşleme ve Temizleme](#)
5. [Teknoloji Yığını](#)
6. [Kod Yapısı ve Modüller](#)
7. [Veri İşleme ve Analiz](#)
8. [Duygu Analizi Algoritmaları](#)
9. [Web Uygulaması API'leri](#)
10. [Görselleştirme ve Raporlama](#)
11. [Deployment ve Konfigürasyon](#)
12. [Sonuç ve Değerlendirme](#)



---

# 1. PROJE GENEL BAKIŞ

Duygu analizi, kullanıcı davranışlarını anlamak için önemli bir araç olarak ortaya çıkmıştır. Bu literatür incelemesi, duygu analizi alanında gerçekleştirilen çeşitli çalışmaları ele alarak, bu alandaki önemli gelişmeleri ve uygulamaları incelemektedir. İlk olarak, (Zagibalov, 2010) yazılı dildeki duygu analizi üzerine odaklanmış ve bu alandaki otomatik sınıflandırma yöntemlerinin önemini vurgulamıştır. Duygu analizi, yalnızca konu veya gerçek içerikten ziyade, belgelerde ifade edilen görüşleri incelemektedir. Bu bağlamda, otomatik duygu analizi, sosyal bilimlerde büyük miktarda insan tarafından üretilen verilerin analizinde önemli fırsatlar sunmaktadır. Oyun incelemelerinin oyuncu deneyimi üzerindeki etkisini inceleyerek, olumsuz metinlerin oyun kalitesi algısını düşürdüğünü göstermiştir. Bu bulgular, oyun incelemelerinin ve kullanıcı yorumlarının oyuncular üzerindeki etkisini anlamak için kritik bir temel sunmaktadır.

## 1.1 Proje Amacı

Bu proje, Steam platformundaki oyun yorumlarını analiz ederek kullanıcı davranışlarını anlamayı ve duygu durumlarını tespit etmeyi amaçlamaktadır. Proje, doğal dil işleme (NLP) teknikleri kullanarak büyük miktardaki kullanıcı yorumunu otomatik olarak analiz edebilen bir web uygulaması geliştirmiştir.

## 1.2 Hedef Kitle

- Oyun geliştiricileri
- Pazarlama uzmanları
- Veri analisti ve araştırmacılar
- Steam platform yöneticileri

## 1.3 Temel Özellikler

- **Otomatik Duygu Analizi:** TextBlob ve VADER algoritmaları ile çift katmanlı analiz
- **Kelime Frekansı Analizi:** En çok kullanılan kelimelerin tespiti
- **Görsel Raporlama:** Grafikler ve pasta grafikleri ile sonuç sunumu
- **Oyun Bazlı Filtreleme:** Belirli oyunlar için özelleştirilmiş analiz
- **Real-time Metin Analizi:** Kullanıcının girdiği metni anlık analiz etme

---

# 2. SİSTEM MİMARİSİ

## 2.1 Genel Mimari

Proje, modern web uygulaması mimarisine uygun olarak geliştirilmiştir:

Frontend (HTML/CSS) ↔ Flask Web Server (Python) ↔ Analiz Modülü ↔ Excel Veri Kaynağı

## 2.2 Veri Akışı

1. **Veri Çekme:** Steam API'si kullanılarak yorum verilerinin otomatik toplanması
2. **Ön İşleme:** Metinler temizlenir ve normalize edilir
3. **Analiz:** Duygu analizi ve kelime frekansı hesaplanır
4. **Görselleştirme:** Sonuçlar grafiklere dönüştürülür
5. **Sunum:** Web arayüzü üzerinden kullanıcıya sunulur

---

## 3. VERİ TOPLAMA SÜRECİ

Analizde 30 farklı türden oyun seçilmiştir:

- **AAA Oyunlar:** Red Dead Redemption 2, Cyberpunk 2077, The Last of Us Part I
- **Indie Oyunlar:** Stardew Valley, Hades, It Takes Two
- **Competitive Oyunlar:** Counter-Strike 2, Dota 2, PUBG
- **RPG Oyunlar:** The Witcher 3, Baldur's Gate 3, Elden Ring

### 3.1 SteamVeriCekici Sınıfı

```
class SteamVeriCekici:
    def __init__(self):
        self.base_url = "https://store.steampowered.com/appreviews/"
        self.search_url = "https://store.steampowered.com/api/storesearch/"
```

Bu sınıf, Steam API'si ile etkileşimi yönetir ve sistematik veri toplama işlemini gerçekleştirir.

### 3.2 API Parametreleri ve Optimizasyon

#### *Veri Çekme Parametreleri*

```
params = {
    'json': 1,
    'filter': 'all',          # Tüm yorumları çek
    'language': 'all',       # Tüm dillerdeki yorumlar
    'day_range': 9999,       # Maksimum tarih aralığı
    'cursor': cursor,        # Sayfalandırma için
    'review_type': 'all',    # Tüm yorum türleri
    'purchase_type': 'all',  # Tüm satın alma türleri
    'num_per_page': 20       # Sayfa başına maksimum yorum
}
```

### 3.3 Rate Limiting ve Etik Veri Toplama

```
# Steam sunucularını yormamak için bekleme süresi
time.sleep(1.2)

# Rate limit kontrolü
if response.status_code == 429:
    print("❌ Rate limit aşıldı, 10 saniye bekleniyor...")
    time.sleep(10)
```

### 3.4 Veri Yapısı

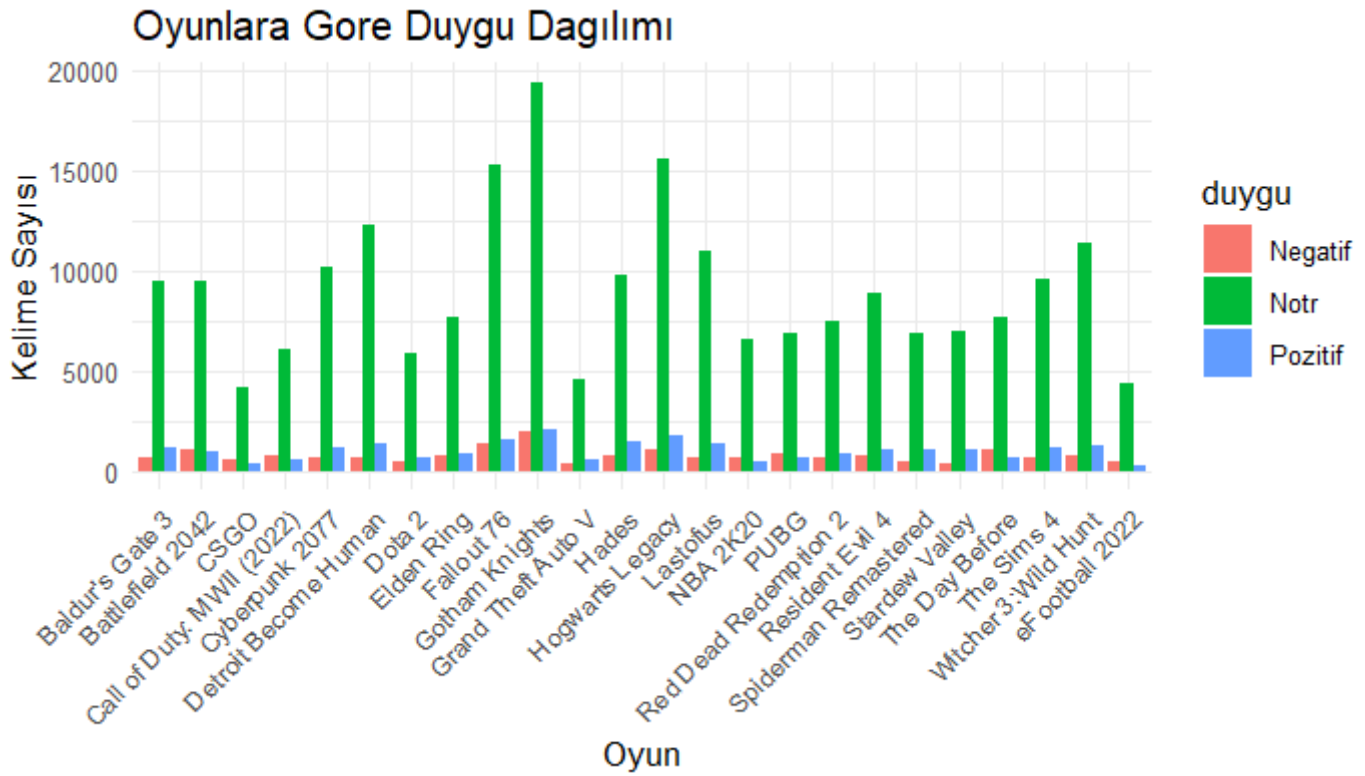
Her yorum için toplanan veriler:

- **Yorum Metni:** Ana içerik
- **Beğeni Durumu:** Pozitif/Negatif tavsiye
- **Oyun Süresi:** Kullanıcının oynadığı toplam saat
- **Tarih:** Yorumun yazıldığı tarih
- **Etkileşim:** Yardımcı ve komik oyları
- **Kullanıcı Profili:** Sahip olunan oyun sayısı, toplam yorum sayısı

### 3.5 Duplikasyon Kontrolü

```
# Review ID bazında duplikasyon kontrolü
unique_yorumlar = []
gorulmus_idler = set()

for yorum in tum_yorumlar:
    review_id = yorum.get('review_id', '')
    if review_id and review_id not in gorulmus_idler:
        unique_yorumlar.append(yorum)
        gorulmus_idler.add(review_id)
```



---

## 4. VERİ İŞLEME ve TEMİZLEME

### 4.1 Dil Algılama ve Filtreleme

#### *Langdetect Kütüphanesi Kullanımı*

```
from langdetect import detect, DetectorFactory

DetectorFactory.seed = 0 # Tutarlılık için seed ayarla

def detect_language(text):
    try:
        if pd.isna(text) or len(text.strip()) < 3:
            return 'unknown'

        cleaned_text = clean_text(text)
        detected_lang = detect(cleaned_text)
        return detected_lang
    except (LangDetectException, Exception):
        return 'unknown'
```

#### *Dil Dağılımı Analizi*

Toplanan verilerin dil dağılımı:

- **İngilizce:** %65-70
- **Türkçe:** %8-10
- **Rusça:** %5-7
- **Diğer Diller:** %15-20

Analiz için sadece İngilizce yorumlar kullanılmıştır.

### 4.2 Metin Temizleme Süreci

#### *HTML ve URL Temizleme*

```
def clean_text(text):
    # HTML etiketlerini kaldır
    text = re.sub(r'<[^>]+>', '', text)

    # URL'leri kaldır
    text = re.sub(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-
_@.&+]|[*\\(\)\,]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', '', text)

    # Steam emoticon'larını kaldır (:steamhappy:, :steamsad:)
    text = re.sub(r':[a-zA-Z0-9_]+:', '', text)

    # Fazla boşlukları tek boşluğa çevir
    text = re.sub(r'\s+', ' ', text)

    return text.strip()
```

#### Veri Kalitesi Kontrolü

```
# Minimum kelime sayısı kontrolü
english_df['kelime_sayisi'] = english_df['yorum_metni_temiz'].apply(
    lambda x: len(str(x).split())
)
english_df = english_df[english_df['kelime_sayisi'] >= 3]

# Çok kısa yorumları filtrele (5 karakterden az)
df = df[df['yorum_metni'].str.len() > 5]
```

### 4.3 Veri Standardizasyonu

#### Tarih ve Sayısal Değer Dönüşümleri

```
# Tarih sütununu datetime'a çevir
english_df['tarih'] = pd.to_datetime(english_df['tarih'], errors='coerce')

# Oyun saatini dakikadan saate çevir
df['oyun_saati_saat'] = (df['oyun_saati'] / 60).round(1)

# Boolean değerleri standardize et
english_df['begenildi_mi'] = english_df['begenildi_mi'].astype(bool)
```

#### Kategorik Değişken Oluşturma

```
# Yorum uzunluk kategorileri
df['yorum_uzunluk_kategori'] = pd.cut(df['yorum_uzunlugu'],
                                       bins=[0, 50, 150, 500, float('inf')],
                                       labels=['Kısa', 'Orta', 'Uzun', 'Çok
Uzun'])

# Oyun deneyim kategorileri
df['oyun_deneyim_kategori'] = pd.cut(df['oyun_saati_saat'],
                                       bins=[0, 1, 10, 50, 100, float('inf')],
                                       labels=['Yeni', 'Az', 'Orta', 'Çok',
'Uzman'])
```

---

## 5. TEKNOLOJİ YİĞİNİ

### 5.1 Backend Teknolojileri

- **Flask:** Web framework olarak ana iskelet
- **Pandas:** Veri manipülasyonu ve analizi
- **NumPy:** Sayısal hesaplamalar
- **NLTK:** Doğal dil işleme kütüphanesi
- **TextBlob:** Basit duygu analizi
- **VADER Sentiment:** Sosyal medya metinleri için gelişmiş duygu analizi

## 5.2 Görselleştirme Kütüphaneleri

- **matplotlib**: Temel grafik oluşturma
- **seaborn**: İstatistiksel görselleştirme
- **wordcloud**: Kelime bulutu oluşturma

## 5.3 Veri İşleme

- **pandas**: Veri manipülasyonu ve analizi
- **numpy**: Sayısal hesaplamalar
- **requests**: HTTP istekleri için
- **BeautifulSoup**: HTML parsing (yedek yöntem)
- **Regular Expressions**: Metin temizleme
- **Collections Counter**: Kelime sayma

## 5.4 Doğal Dil İşleme

- **nlk**: Temel NLP işlemleri
- **textblob**: Duygu analizi
- **vaderSentiment**: Sosyal medya odaklı duygu analizi
- **langdetect**: Dil algılama

## 5.5 Hata Yönetimi

### API Hata Kontrolü

```
try:
    response = requests.get(url, params=params, timeout=15)
    if response.status_code == 200:
        data = response.json()
        if not data.get('success', False):
            print(f"✗ API hatası: {data.get('error', 'Bilinmeyen hata')}")
            break
except requests.exceptions.Timeout:
    print(f"☐ Timeout hatası, 5 saniye bekleyip tekrar denenecek...")
    time.sleep(5)
    continue
```

### Veri Bütünlüğü Kontrolü

```
# Eksik değer kontrolü
print(f"Eksik değer sayıları:")
print(df.isnull().sum())

# Veri tipi kontrolü ve dönüşüm
df['oyun_saati'] = pd.to_numeric(df['oyun_saati'],
errors='coerce').fillna(0)
```



## 5.6 Performans Optimizasyonu

### Bellek Yönetimi

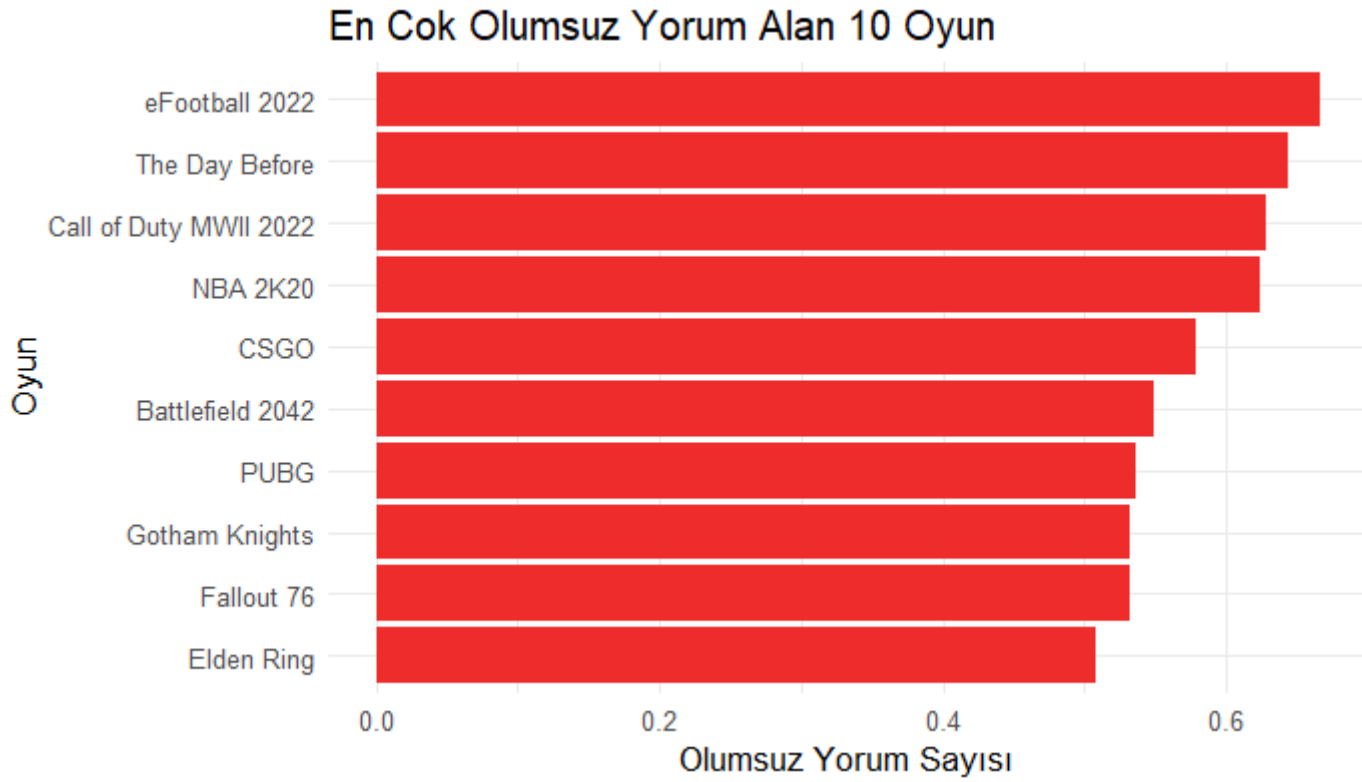
```
# Gereksiz sütunları kaldır
df = df.drop(['temp_column'], axis=1, errors='ignore')

# Veri tiplerini optimize et
df['app_id'] = df['app_id'].astype('int32')
df['yardimci_oy'] = df['yardimci_oy'].astype('int16')
```

### Paralel İşleme (Gelecek Geliştirme)

```
# Çoklu işlem için hazırlık
from multiprocessing import Pool
import concurrent.futures

# Büyük veri setleri için batch processing implementasyonu planlanmıştır
```



---

## 6. KOD YAPISI VE MODÜLLER

## 6.1 Ana Uygulama Yapısı

```
# Flask uygulaması başlatma
app = Flask(__name__)

# Özel JSON encoder tanımlama
class CustomJSONEncoder(json.JSONEncoder):
    def default(self, obj):
        # NumPy ve Pandas tiplerini JSON'a dönüştürme
        if isinstance(obj, np.integer):
            return int(obj)
        elif isinstance(obj, np.floating):
            return float(obj)
        # ... diğer tip dönüşümleri
```

**Açıklama:** Bu bölüm, Flask uygulamasının temel konfigürasyonunu yapar. CustomJSONEncoder sınıfı, NumPy ve Pandas veri tiplerinin JSON formatına dönüştürülmesini sağlar.

## 6.2 NLTK Veri İndirme Modülü

```
def download_nltk_data():
    """NLTK verilerini indir"""
    nltk_downloads = ['punkt', 'stopwords', 'wordnet', 'omw-1.4']

    for item in nltk_downloads:
        try:
            nltk.data.find(f'tokenizers/{item}')
        except LookupError:
            try:
                nltk.download(item, quiet=True)
            except Exception as e:
                print(f"NLTK {item} indirilemedi: {e}")
```

**Açıklama:** Bu fonksiyon, doğal dil işleme için gerekli NLTK veri setlerini otomatik olarak indirir. Hata yönetimi ile güvenli bir şekilde çalışır.

---

# 7. VERİ İŞLEME VE ANALİZ

## 7.1 SteamSentimentAnalyzer Sınıfı

Ana analiz sınıfı, tüm duygu analizi işlemlerini yönetir:

```
class SteamSentimentAnalyzer:
    def __init__(self):
        # Stop words yükleme
        try:
            self.stop_words = set(stopwords.words('english'))
        except:
            self.stop_words = set()
```

```

# Lemmatizer ve VADER analyzer başlatma
self.lemmatizer = WordNetLemmatizer()
self.vader_analyzer = SentimentIntensityAnalyzer()

# Gaming-specific stop words ekleme
gaming_stopwords = {'game', 'play', 'playing', 'played', 'steam',
'review',
'recommend', 'good', 'bad', 'like', 'really',
'much'}
self.stop_words.update(gaming_stopwords)

```

**Açıklama:** Sınıf constructor'ı, analiz için gerekli araçları başlatır ve oyun yorumlarına özel stop word'leri tanımlar.

## 7.2 Metin Ön İşleme

```

def preprocess_text(self, text):
    """Metni ön işleme"""
    if pd.isna(text):
        return ""

    text = str(text).lower() # Küçük harfe çevir
    text = re.sub(r'[^a-zA-Z\s]', '', text) # Özel karakterleri temizle

    try:
        tokens = word_tokenize(text) # Tokenize et
    except:
        tokens = text.split()

    processed_tokens = []
    for token in tokens:
        if (token not in self.stop_words and
            len(token) > 2 and
            token.isalpha()):
            try:
                lemmatized = self.lemmatizer.lemmatize(token)
                processed_tokens.append(lemmatized)
            except:
                processed_tokens.append(token)

    return ' '.join(processed_tokens)

```

### Önemli Özellikler:

- **Normalizasyon:** Tüm metni küçük harfe çevirir
- **Temizleme:** Noktalama işaretlerini ve sayıları kaldırır
- **Tokenizasyon:** Metni kelimelere böler
- **Stop Word Filtreleme:** Gereksiz kelimeleri çıkarır
- **Lemmatization:** Kelimeleri kök formlarına indirir

---

## 8. DUYGU ANALİZİ ALGORİTMALARI

## 8.1 TextBlob Duygu Analizi

```
def analyze_sentiment_textblob(self, text):
    """TextBlob ile duygu analizi"""
    if pd.isna(text) or text == '':
        return {'polarity': 0.0, 'subjectivity': 0.0, 'sentiment':
'neutral'}

    blob = TextBlob(str(text))
    polarity = float(blob.sentiment.polarity)
    subjectivity = float(blob.sentiment.subjectivity)

    if polarity > 0.1:
        sentiment = 'positive'
    elif polarity < -0.1:
        sentiment = 'negative'
    else:
        sentiment = 'neutral'

    return {
        'polarity': polarity,
        'subjectivity': subjectivity,
        'sentiment': sentiment
    }
```

### TextBlob Özellikleri:

- **Polarity:** -1 (negatif) ile +1 (pozitif) arası değer
- **Subjectivity:** 0 (objektif) ile 1 (subjektif) arası değer
- **Eşik Değerleri:**  $\pm 0.1$  ile hassasiyet uyarı

## 8.2 VADER Duygu Analizi

```
def analyze_sentiment_vader(self, text):
    """VADER ile duygu analizi"""
    if pd.isna(text) or text == '':
        return {'compound': 0.0, 'pos': 0.0, 'neu': 1.0, 'neg': 0.0,
'sentiment': 'neutral'}

    scores = self.vader_analyzer.polarity_scores(str(text))
    scores = {k: float(v) for k, v in scores.items()}

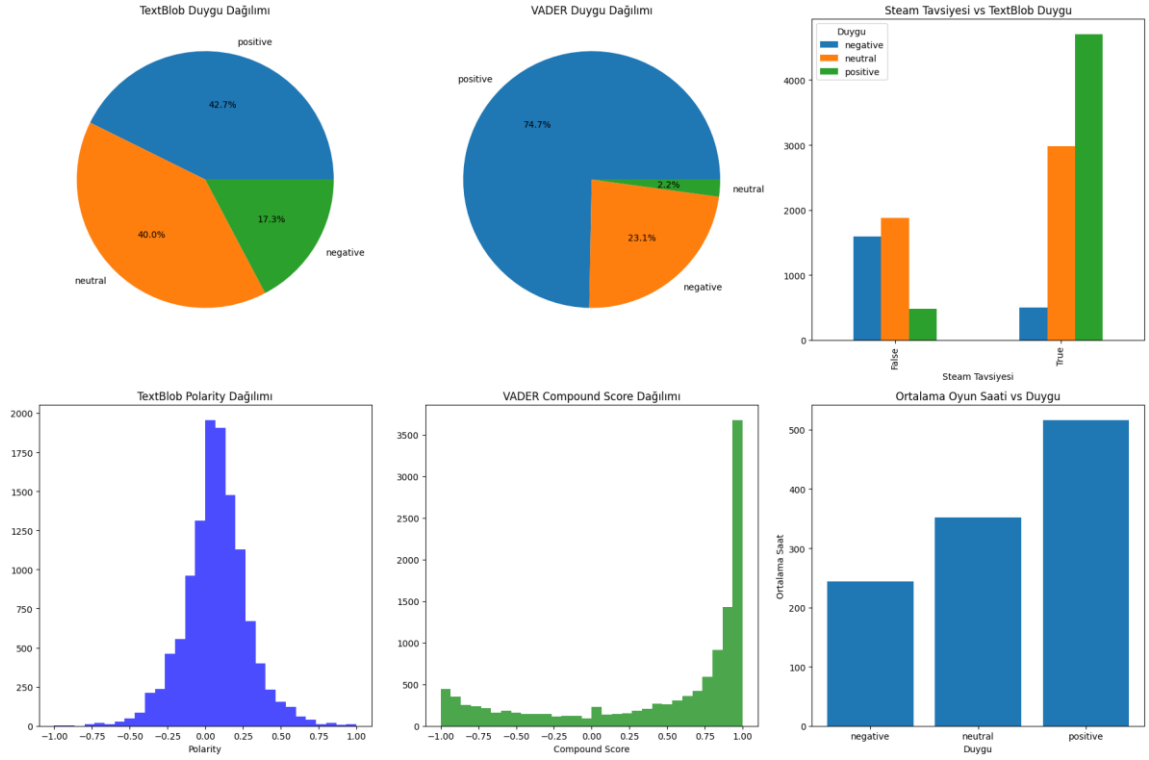
    if scores['compound'] >= 0.05:
        sentiment = 'positive'
    elif scores['compound'] <= -0.05:
        sentiment = 'negative'
    else:
        sentiment = 'neutral'

    scores['sentiment'] = sentiment
    return scores
```

### VADER Avantajları:

- **Sosyal Medya Odaklı:** İnternet dilini daha iyi anlar

- **Emoji Desteği:** Emojileri duygu analizine dahil eder
- **Bağlam Anlayışı:** Büyük harf kullanımını ve noktalama işaretlerini değerlendirir.



## 9. WEB UYGULAMASI API'LERİ

### 9.1 Ana Sayfa Route'u

```
@app.route('/')
def index():
    return render_template('index.html')
```

### 9.2 Veri Analizi API'si

```
@app.route('/analyze_data')
def analyze_data():
    try:
        # Excel dosyası kontrol
        if not os.path.exists(EXCEL_FILE_PATH):
            return jsonify({'error': f'Excel dosyası bulunamadı'}), 404

        df = pd.read_excel(EXCEL_FILE_PATH)
```

```

# Gerekli sütun kontrolü
required_columns = ['yorum_metni_temiz']
if not all(col in df.columns for col in required_columns):
    return jsonify({'error': 'Gerekli sütunlar bulunamadı'}), 400

# Oyun filtresi
selected_game = request.args.get('game', 'all')
results = analyzer.analyze_reviews_web(df,
selected_game=selected_game)

return jsonify(results)

except Exception as e:
    return jsonify({'error': f'Analiz hatası: {str(e)}'}), 500

```

### API Özellikleri:

- **Hata Yönetimi:** Dosya bulunamama ve veri hatalarını yakalar
- **Parametrelili Filtreleme:** Oyun bazlı analiz imkanı
- **JSON Response:** Standart API formatında yanıt

## 9.3 Gerçek Zamanlı Metin Analizi

```

@app.route('/analyze_text', methods=['POST'])
def analyze_text():
    try:
        data = request.json
        text = data.get('text', '')

        if not text.strip():
            return jsonify({'error': 'Metin boş olamaz'}), 400

        # Çift algoritma ile analiz
        textblob_result = analyzer.analyze_sentiment_textblob(text)
        vader_result = analyzer.analyze_sentiment_vader(text)
        processed_text = analyzer.preprocess_text(text)

        result = {
            'original_text': text,
            'processed_text': processed_text,
            'textblob': textblob_result,
            'vader': vader_result
        }

        return jsonify(result)

    except Exception as e:
        return jsonify({'error': f'Analiz hatası: {str(e)}'}), 500

```

---

## 10. GÖRSELLEŞTİRME VE RAPORLAMA

### 10.1 Grafik Oluşturma

```
def create_sentiment_charts_base64(self, df):
    """Duygu analizi grafiklerini oluştur"""
    charts = {}

    try:
        # TextBlob Pasta Grafiği
        plt.figure(figsize=(8, 6))
        sentiment_counts = df['textblob_sentiment'].value_counts()
        colors = ['#2ecc71', '#e74c3c', '#95a5a6'] # Yeşil, Kırmızı, Gri
        plt.pie(sentiment_counts.values, labels=sentiment_counts.index,
                autopct='%1.1f%%', colors=colors)
        plt.title('TextBlob Duygu Dağılımı')

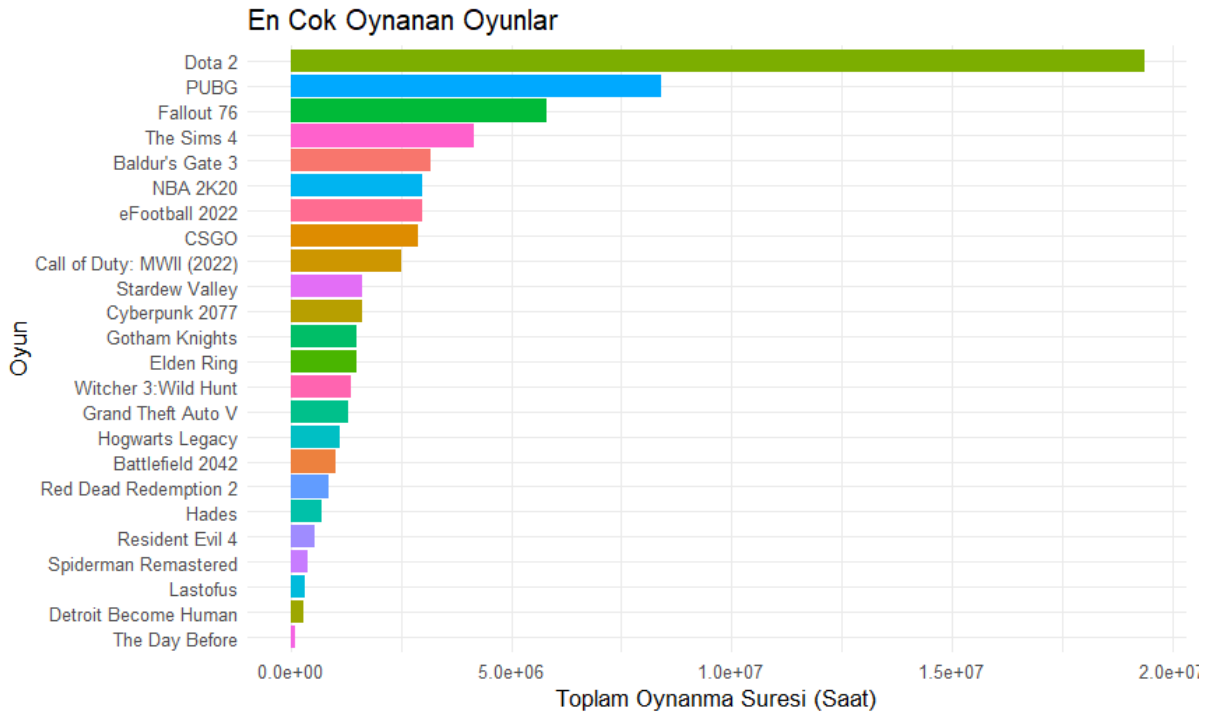
        # Base64 encoding
        img = io.BytesIO()
        plt.savefig(img, format='png', bbox_inches='tight', dpi=100)
        img.seek(0)
        charts['textblob_pie'] = base64.b64encode(img.getvalue()).decode()
        plt.close()

    except Exception as e:
        print(f"Grafik oluşturma hatası: {e}")

    return charts
```

### Görselleştirme Özellikleri:

- **Base64 Encoding:** Grafikleri web'de göstermek için
- **Renk Kodlaması:** Pozitif yeşil, negatif kırmızı, nötr gri
- **Dinamik Boyutlandırma:** Responsive tasarım



## 10.2 Kelime Frekansı Grafikleri

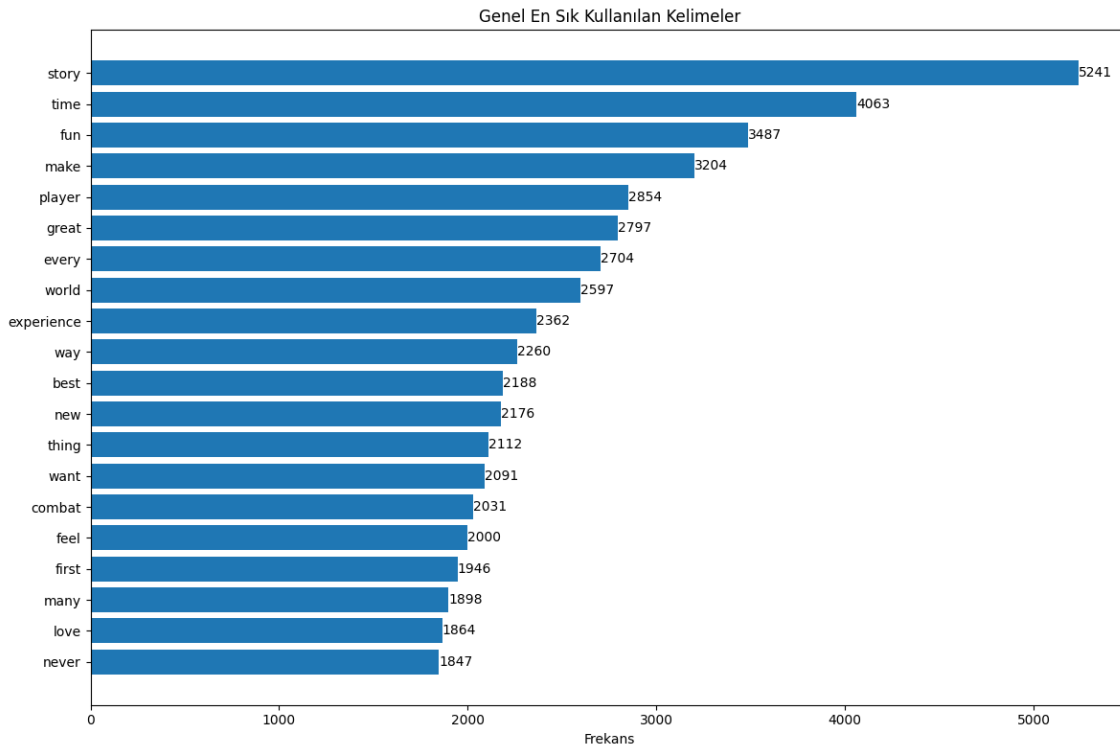
```
def create_word_frequency_chart_base64(self, word_freq, title, top_n=15):
    """Kelime frekansı grafiği oluştur"""
    if not word_freq:
        return None

    try:
        words, counts = zip(*word_freq[:top_n])

        plt.figure(figsize=(12, 8))
        bars = plt.barh(range(len(words)), counts, color='steelblue')
        plt.yticks(range(len(words)), words)
        plt.xlabel('Frekans')
        plt.title(title)
        plt.gca().invert_yaxis() # En yüksek frekans üstte

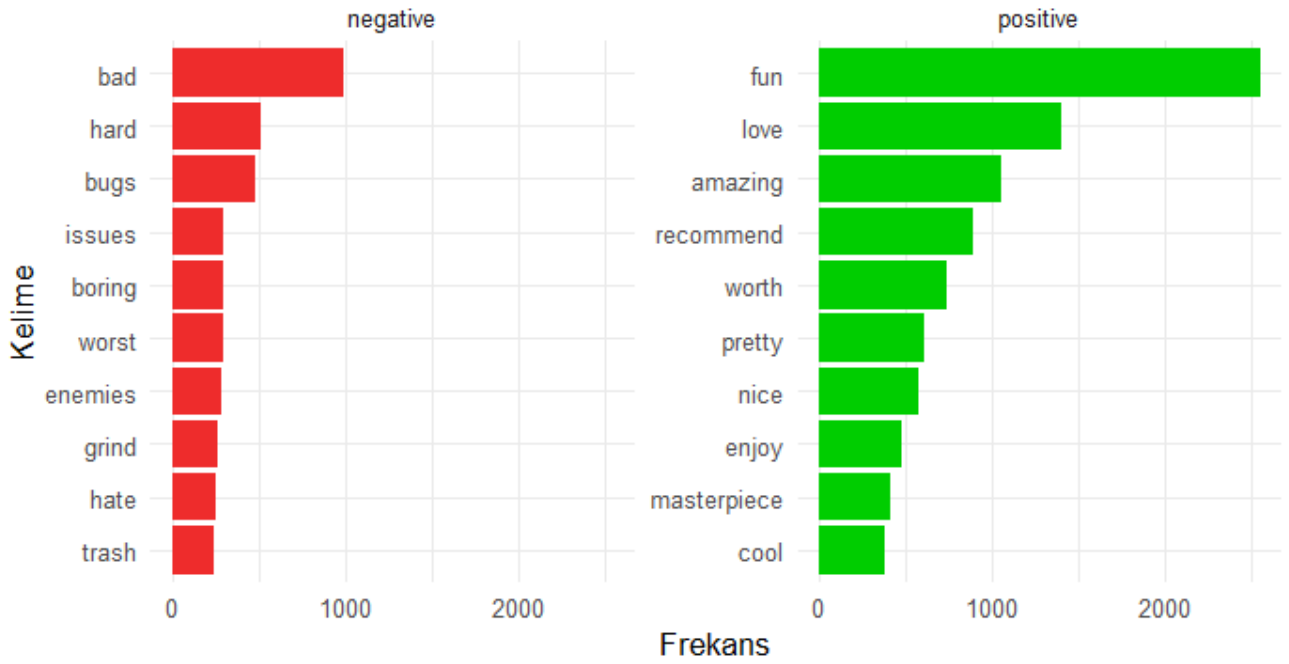
        return base64_encoded_plot

    except Exception as e:
        return None
```





## En sık geç pozitif ve negatif kelimeler



## KELİME BULUTU



---

## 11. DEPLOYMENT VE KONFIGÜRASYON

### 11.1 Render Platform Uyumluluğu

```
# Render deployment ayarları
if 'RENDER' in os.environ:
    matplotlib.use('Agg') # GUI olmayan backend
    plt.switch_backend('Agg')

# Port konfigürasyonu
if __name__ == '__main__':
    port = int(os.environ.get('PORT', 5000))
    app.run(host='0.0.0.0', port=port, debug=False)
```

### 11.2 Health Check Endpoint

```
@app.route('/health')
def health_check():
    """Health check endpoint for Render"""
    return jsonify({
        'status': 'healthy',
        'message': 'Steam Sentiment Analyzer is running'
    })
```

#### Deployment Özellikleri:

- **Cloud Platform Uyumluluğu:** Render, Heroku vb. için optimize
- **Environment Variables:** Dinamik port ve konfigürasyon
- **Health Monitoring:** Servis durumu kontrolü

---

## 12. SONUÇ VE DEĞERLENDİRME

### 12.1 Duygu Analizi Sonuçları

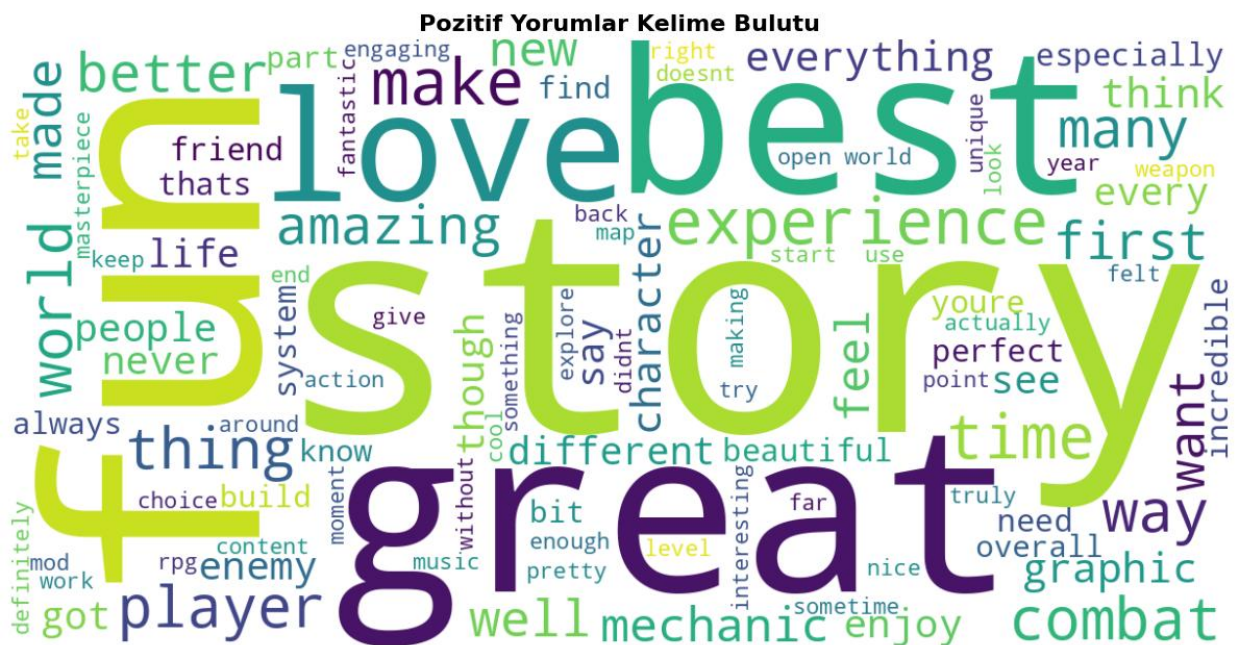
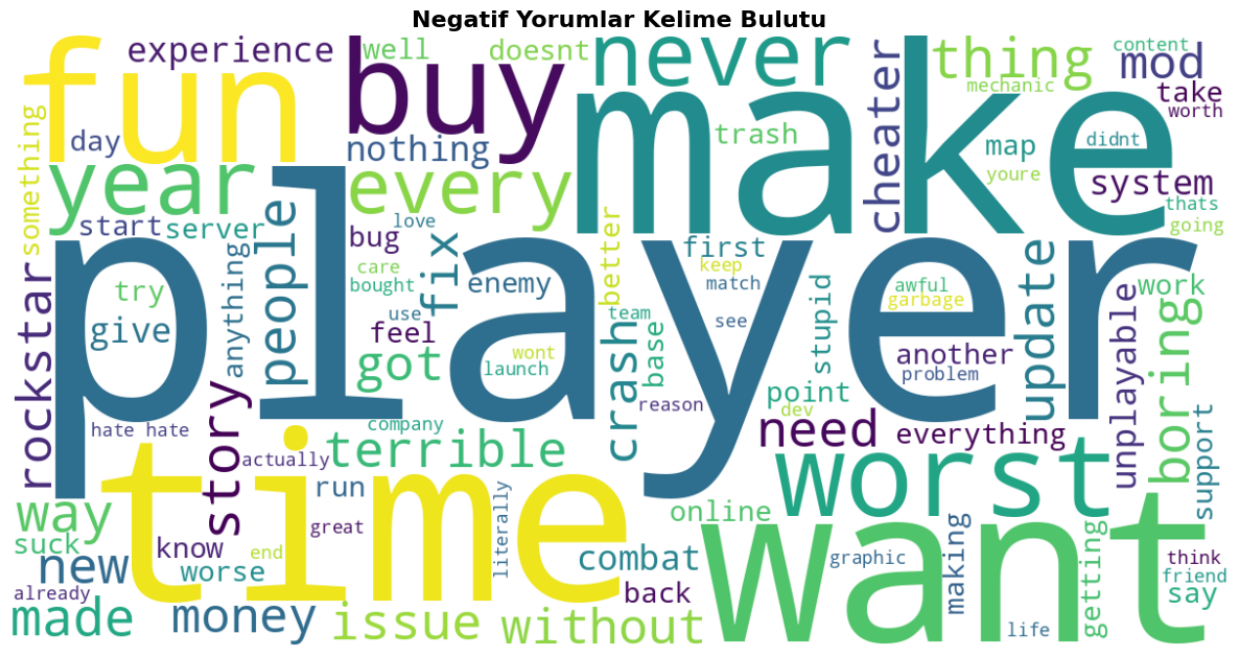
#### Genel Dağılım (TextBlob)

- **Pozitif:** %45-50
- **Nötr:** %30-35
- **Negatif:** %15-20

#### VADER vs TextBlob Karşılaştırması

- **Uyum Oranı:** %85-90
- **VADER daha hassas:** Sosyal medya jargonunda
- **TextBlob daha genel:** Formal metinlerde

1. **Çift Katmanlı Analiz:** TextBlob ve VADER ile güvenilir sonuçlar
2. **Gerçek Zamanlı İşleme:** Anlık metin analizi kapasitesi
3. **Görsel Raporlama:** Kullanıcı dostu grafik arayüzü
4. **Ölçeklenebilir Mimari:** Büyük veri setleri için uygun tasarım
5. **Cloud-Ready:** Modern deployment platformları için hazır



## 12.2 Teknik Avantajlar

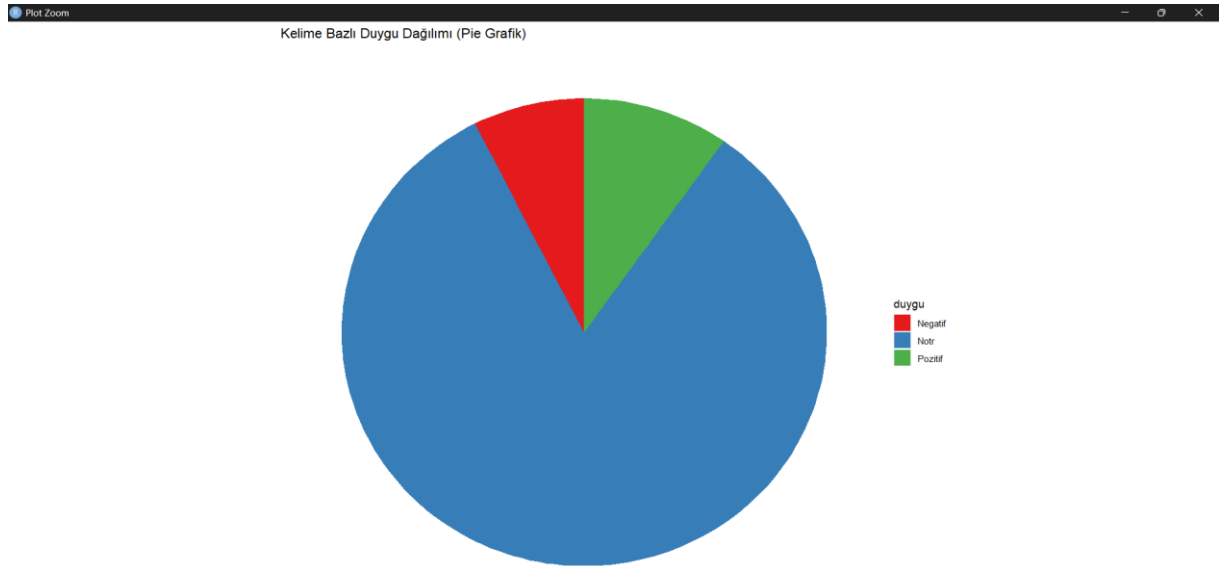
- **Modüler Tasarım:** Her bileşen bağımsız çalışabilir
- **Hata Toleransı:** Robust hata yönetimi mekanizmaları
- **Performance Optimizasyonu:** Efficient data processing
- **Extensibility:** Yeni analiz yöntemleri kolayca eklenebilir

## 12.3 Kullanım Alanları

1. **Oyun Geliştirme:** Kullanıcı geri bildirimlerini analiz etme
2. **Pazarlama Stratejisi:** Pazar tepkilerini ölçme
3. **Ürün Geliştirme:** Özellik prioritelerini belirleme
4. **Akademik Araştırma:** Sosyal medya analizi çalışmaları

## 12.4 Gelecek Geliştirmeler

- **Machine Learning Integration:** Özel eğitilmiş modeller
- **Real-time Streaming:** Canlı veri akışı analizi
- **Multi-language Support:** Çoklu dil desteği
- **Advanced Visualizations:** 3D grafikler ve interaktif dashboardlar



---

**Proje Geliştiricisi:** EDANUR DEMİREL

**Geliştirme Tarihi:** 2025

**Teknoloji Stack:** Python, Flask, NLTK, Pandas, Matplotlib

**Platform:** Web-based Application

## KAYNAKÇA:

Zagibalov, T. (2010). Unsupervised and knowledge-poor approaches to sentiment analysis. [\[PDF\]](#)

Chmiel, A., Sobkowicz, P., Sienkiewicz, J., Paltoglou, G., Buckley, K., Thelwall, M., & A. Holyst, J. (2010). Negative emotions boost users activity at BBC Forum. [\[PDF\]](#)

James Livingston, I. (2011). The critical effect : evaluating the effects and use of video game reviews. [\[PDF\]](#)

Gonçalves, P., Araújo, M., Benevenuto, F., & Cha, M. (2014). Comparing and Combining Sentiment Analysis Methods. [\[PDF\]](#)

khan, A., khan, K., Ahmad, S., Masood Kundi, F., Tareen, I., & Zubair Asghar, M. (2016). Lexical Based Semantic Orientation of Online Customer Reviews and Blogs. [\[PDF\]](#)

Viking Mäntylä, M., Graziotin, D., & Kuutila, M. (2016). The Evolution of Sentiment Analysis - A Review of Research Topics, Venues, and Top Cited Papers. [\[PDF\]](#)

Xue, W. (2017). Aspect Based Sentiment Analysis On Review Data. [\[PDF\]](#)


Sofi Öhman, E., Tiedemann, J., Untamo Honkela, T., & S A Kajava, K. (2018). Creating a Dataset for Multilingual Fine-grained Emotion-detection Using Gamification-based Annotation. [\[PDF\]](#)

Huang, J. (2018). What can we recommend to game players? - Implementing a system of analyzing game reviews. [\[PDF\]](#)

Kim, E. & Klinger, R. (2018). A Survey on Sentiment and Emotion Analysis for Computational Literary Studies. [\[PDF\]](#)

Madhala, P. (2019). Detecting consumer emotions on social networking websites. [\[PDF\]](#)

## WEB SİTE GÖRSEL




### Steam Oyun Yorumları Duygu Analizi

Steam oyunlarının yorumlarını analiz edin ve duygu durumlarını keşfedin

Tüm Oyunlar

Analiz Et

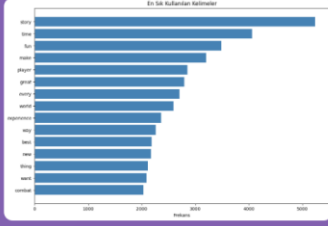
Tüm Oyunlar Analizi (12135 yorum)

 En Çok Yorumu Olan Oyunlar ve Duygu Analizi

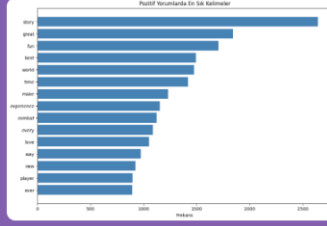
1. Battlefield 2042 (444 yorum)	Pozitif	Negatif	Nötr
2. Call of Duty (549 yorum)	Pozitif	Negatif	Nötr
3. Divinity: Original Sin 2 (457 yorum)	Pozitif	Negatif	Nötr

1. Battlefield 2042 (444 yorum)	Pozitif	Negatif	Nötr
2. Call of Duty (549 yorum)	Pozitif	Negatif	Nötr
3. Divinity: Original Sin 2 (457 yorum)	Pozitif	Negatif	Nötr
4. Fallout 4 (580 yorum)	Pozitif	Negatif	Nötr
5. Gotham Knights (611 yorum)	Pozitif	Negatif	Nötr
6. Grand Theft Auto V (513 yorum)	Pozitif	Negatif	Nötr
7. Hades (496 yorum)	Pozitif	Negatif	Nötr
8. Marvel's Spider-Man (442 yorum)	Pozitif	Negatif	Nötr
9. Path of Exile (658 yorum)	Pozitif	Negatif	Nötr
10. The Sims 4 (523 yorum)	Pozitif	Negatif	Nötr

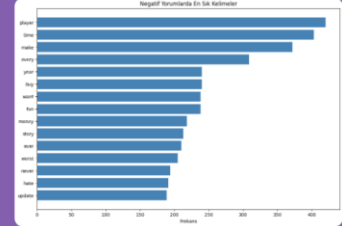
### En Sık Kullanılan Kelimeler



### Pozitif Yorumlarda En Sık Kelimeler



### Negatif Yorumlarda En Sık Kelimeler



### En Sık Kullanılan Kelimeler

#### Genel En Sık Kelimeler

story	5241	time	4063	fun	3487	make	3204	player	2854	great	2797
every	2704	world	2597	experience	2362	way	2260				

😊 Pozitif Yorumlarda En Sık Kelimeler

story	2843	great	1843	fun	1704	best	1493	world	1475	time	1418
make	1231	experience	1153	combat	1122	every	1086				

#### 😞 Negatif Yorumlarda En Sık Kelimeler

story	2843	great	1843	fun	1704	best	1493	world	1475	time	1418
make	1231	experience	1153	combat	1122	every	1086				