1. Giriş

Bu proje, oyun oynama verilerini analiz ederek oyuncuların **EngagementLevel** (Katılım Seviyesi) sınıflandırmasını öngörmeyi amaçlamaktadır. Veri seti, oyuncuların demografik bilgileri, oyun alışkanlıkları ve başarı metriklerini içermektedir. Proje, veri ön işleme, keşifsel veri analizi (EDA), özellik mühendisliği ve beş farklı makine öğrenmesi modelinin performans karşılaştırmasını kapsamaktadır.

2. Proje Yapısı

Proje, modüler bir yaklaşımla aşağıdaki dosyalardan oluşmaktadır:

- data_preprocessing.py: Veri yükleme ve temizleme
- exploratory_analysis.py: Keşifsel veri analizi ve görselleştirme
- feature_engineering.py: Özellik oluşturma ve dönüşüm
- model_training.py: Model eğitimi ve değerlendirme
- main.py: Ana çalıştırma dosyası

3. Veri Ön İşleme

Veri seti online_gaming_behavior_dataset.csv dosyasından yüklenmiştir. Temizleme ve hazırlık adımları aşağıdaki gibi gerçekleştirilmiştir:

```
def clean_data(df):
    """Veri setini temizler: eksik değerleri ve veri tiplerini düzenler"""
    print("Eksik değerler:\n", df.isnull().sum())

    categorical_cols = ['Gender', 'Location', 'GameGenre', 'GameDifficulty',
'EngagementLevel']
    for col in categorical_cols:
        df[col] = df[col].astype('category')

    numerical_cols = ['Age', 'PlayTimeHours', 'InGamePurchases',
'SessionsPerWeek',
                      'AvgSessionDurationMinutes', 'PlayerLevel',
'AchievementsUnlocked']
    for col in numerical_cols:
        df[col] = pd.to_numeric(df[col], errors='coerce')

    return df
```

```
def remove_outliers(df):
    """IQR yöntemiyle aykırı değerleri kaldırır"""
    numerical_cols = df.select_dtypes(include=[np.number]).columns
    for col in numerical_cols:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
    return df
```

- **Eksik Değerler**: Eksik değerler kontrol edilmiş ve sayısal sütunlar için coerce ile NaN değerlerine dönüştürülmüştür.
- **Veri Tipleri**: Kategorik değişkenler category tipine, sayısal değişkenler numeric tipine çevrilmiştir.
- **Aykırı Değerler**: IQR yöntemiyle aykırı değerler kaldırılmıştır.

4. Keşifsel Veri Analizi (EDA)

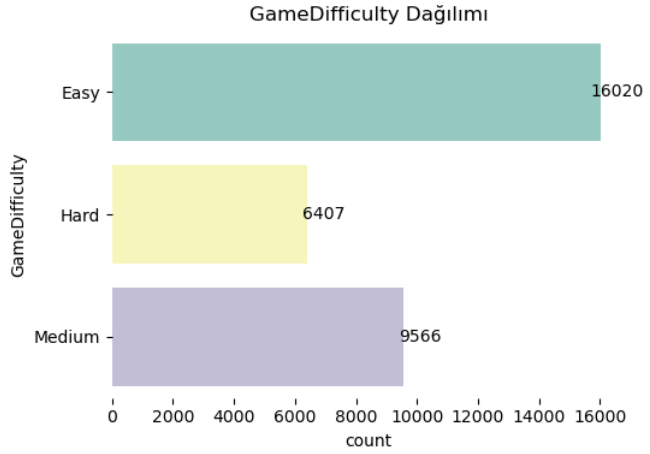
EDA, veri setinin yapısını ve dağılımlarını anlamak için gerçekleştirilmiştir. Görselleştirmeler hem kategorik hem de sayısal değişkenler için oluşturulmuştur.

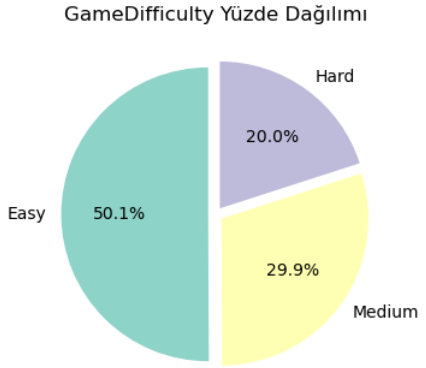
4.1 Kategorik Değişken Analizi

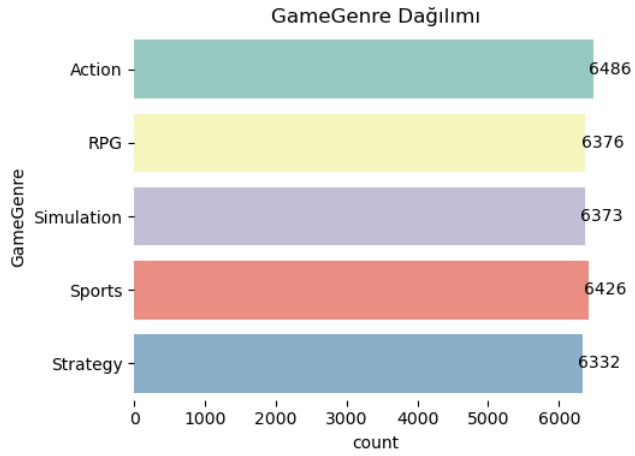
Kategorik değişkenler için çubuk ve pasta grafikler kullanılmıştır.

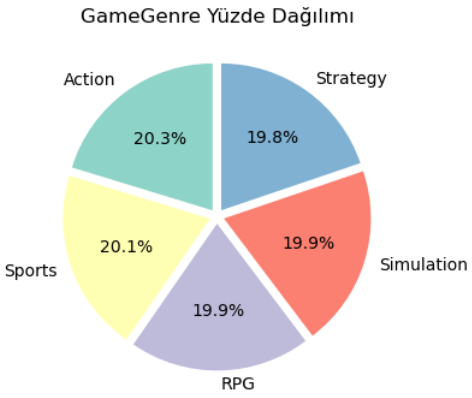
```
def plot_categorical_distribution(df, column_name):
    """Kategorik değişkenlerin dağılımını görselleştirir"""
    plt.figure(figsize=(10, 4))

    plt.subplot(1, 2, 1)
    ax = sns.countplot(y=column_name, data=df, palette='Set3')
    plt.title(f'{column_name} Dağılımı')
    for p in ax.patches:
        ax.annotate(f'{int(p.get_width())}',
                    (p.get_width(), p.get_y() + p.get_height() / 2),
                    ha='center', va='center', xytext=(10, 0),
textcoords='offset points')
    sns.despine(left=True, bottom=True)

    plt.subplot(1, 2, 2)
    df[column_name].value_counts().plot.pie(
        autopct='%1.1f%%',
```

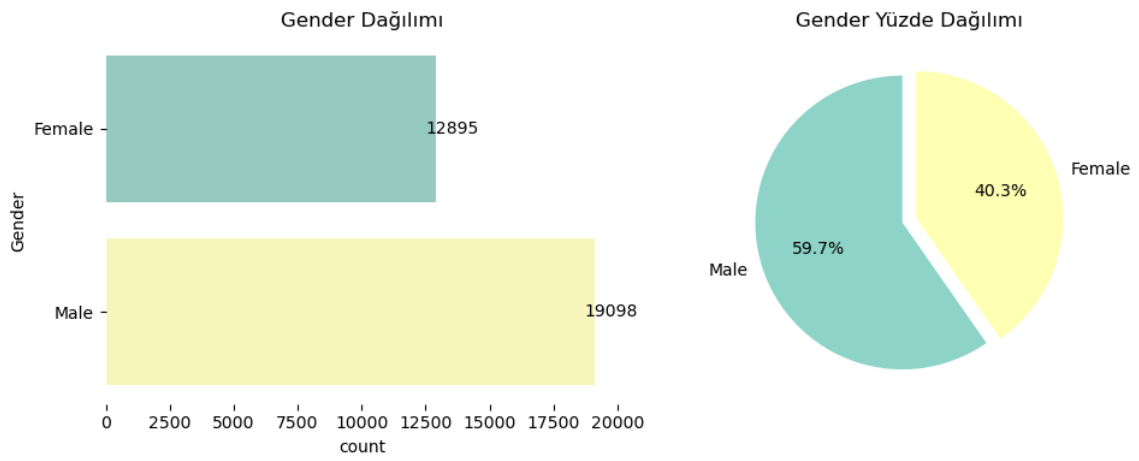
```
        colors=sns.color_palette('Set3'),
        startangle=90,
        explode=[0.05] * df[column_name].nunique()
    )
    plt.title(f'{column_name} Yüzde Dağılımı')
    plt.ylabel('')

    plt.tight_layout()
    plt.savefig(f'{column_name}_distribution.png')
    plt.close()
```

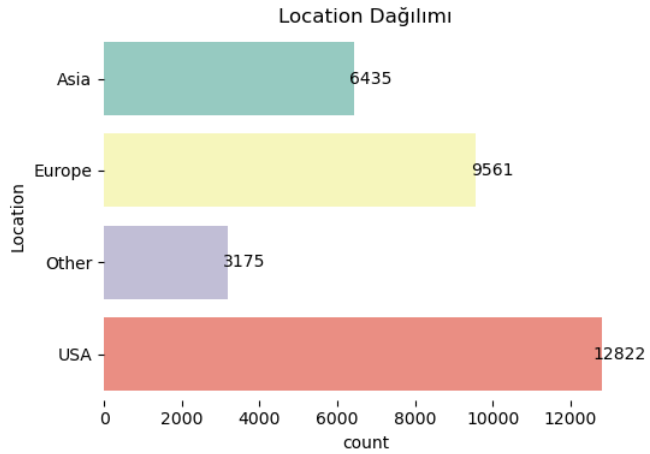


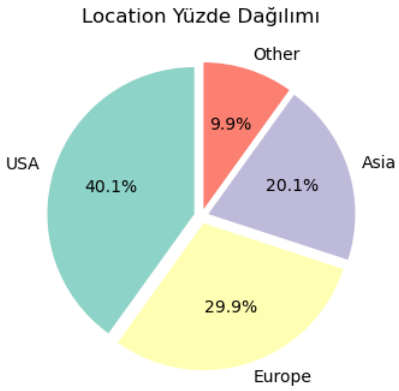








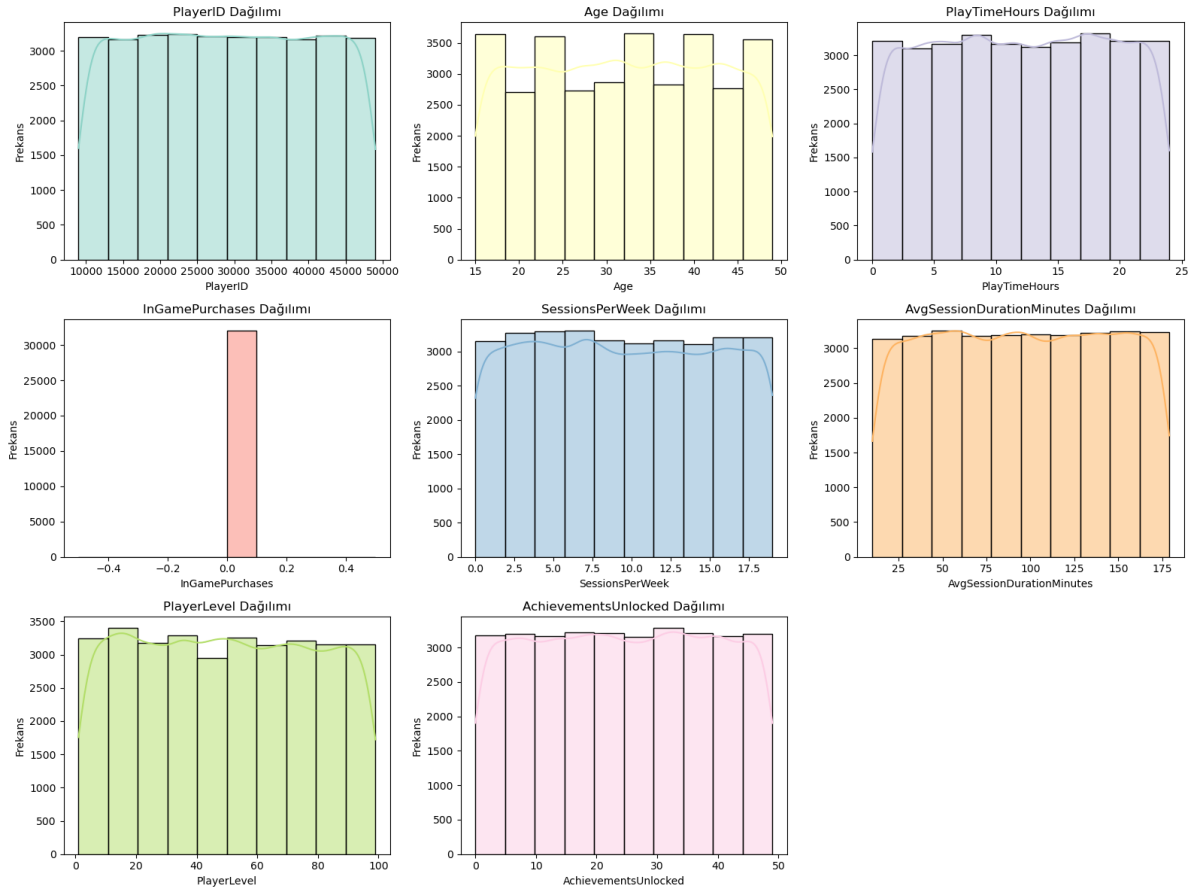




4.2 Sayısal Değişken Analizi

Sayısal değişkenlerin dağılımları histogram ve KDE (yoğunluk tahmini) ile görselleştirilmiştir.

```python
def visualize_distributions(df):
    """Sayısal değişkenler için dağılım grafikleri oluşturur"""
    numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
    set3_colors = sns.color_palette("Set3", len(numerical_cols))

    plt.figure(figsize=(16, 12))
    for i, column in enumerate(numerical_cols, 1):
        plt.subplot(3, 3, i)
        sns.histplot(df[column], kde=True, bins=10, color=set3_colors[i-1])
        plt.title(f'{column.replace("_", " ")} Dağılımı')
        plt.xlabel(column.replace('_', ' '))
        plt.ylabel('Frekans')

    plt.tight_layout()
    plt.savefig('numerical_distributions.png')
    plt.close()
```



5. Özellik Mühendisliği

Veri setine yeni özellikler eklenmiş ve kategorik değişkenler kodlanmıştır.

```
def create_features(df):
    """Mevcut verilerden yeni özellikler oluşturur"""
    # Toplam oyun süresini dakikaya çevir
    df['TotalPlayTimeMinutes'] = df['PlayTimeHours'] * 60

    # Oturum başına ortalama başarı
    df['AchievementsPerSession'] = df['AchievementsUnlocked'] /
(df['SessionsPerWeek'] + 1)

    return df

def encode_categorical(df):
    """Kategorik değişkenleri kodlar"""
    categorical_cols = ['Gender', 'Location', 'GameGenre', 'GameDifficulty']
    df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
    return df_encoded

def scale_features(df):
```

```
    """Sayısal özellikleri ölçeklendirir"""
    numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
    scaler = StandardScaler()
    df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
    return df, scaler
```

- **Yeni Özellikler**:
  - TotalPlayTimeMinutes: Toplam oyun süresini dakikaya çevirir.
  - AchievementsPerSession: Oturum başına başarı oranını hesaplar.
- **Kodlama**: Kategorik değişkenler one-hot encoding ile sayısallaştırılmıştır.
- **Ölçeklendirme**: Sayısal özellikler standartlaştırılmıştır.

6. Modelleme ve Değerlendirme

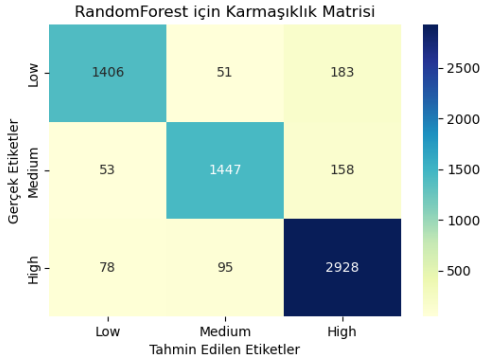
Beş farklı makine öğrenmesi modeli eğitilmiş ve performansları karşılaştırılmıştır:

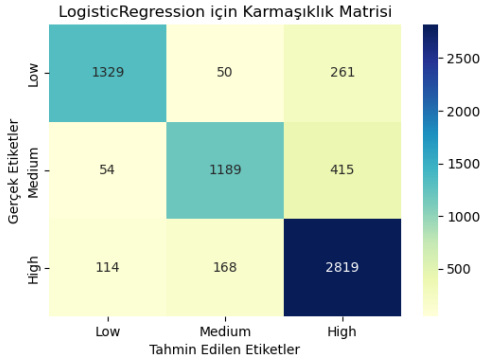
1. Random Forest
2. Logistic Regression
3. SVM
4. Gradient Boosting
5. KNN

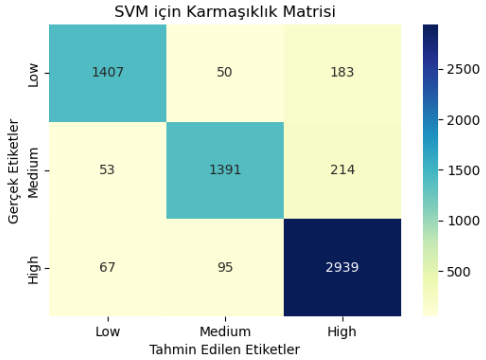
```
def train_models(X_train, y_train):
    """Farklı modelleri eğitir ve bir sözlük olarak döndürür"""
    models = {
        'RandomForest': RandomForestClassifier(n_estimators=100,
random_state=42),
        'LogisticRegression': LogisticRegression(max_iter=1000,
random_state=42),
        'SVM': SVC(random_state=42),
        'GradientBoosting': GradientBoostingClassifier(n_estimators=100,
random_state=42),
        'KNN': KNeighborsClassifier(n_neighbors=5)
    }

    trained_models = {}
    for name, model in models.items():
        print(f"\n{name} modeli eğitiliyor...")
        model.fit(X_train, y_train)
        trained_models[name] = model

    return trained_models

def evaluate_models(models, X_test, y_test):
    """Tüm modellerin performansını değerlendirir ve görselleştirir"""
    model_results = []
```

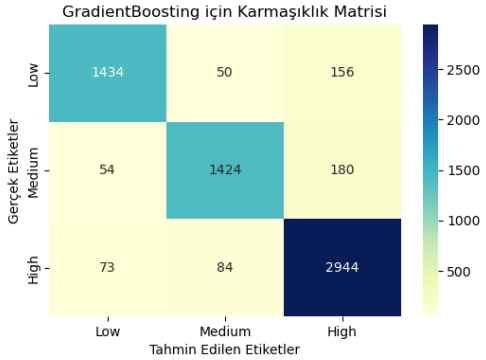
```
    for name, model in models.items():
        print(f"\n{name} Model Değerlendirmesi:")
        y_pred = model.predict(X_test)

        accuracy = accuracy_score(y_test, y_pred)
        print("Sınıflandırma Raporu:")
        print(classification_report(y_test, y_pred))

        cm = confusion_matrix(y_test, y_pred)
        plt.figure(figsize=(6, 4))
        sns.heatmap(cm, annot=True, fmt="d", cmap="YlGnBu",
                    xticklabels=['Low', 'Medium', 'High'],
                    yticklabels=['Low', 'Medium', 'High'])
        plt.title(f"{name} için Karmaşıklık Matrisi")
        plt.xlabel("Tahmin Edilen Etiketler")
        plt.ylabel("Gerçek Etiketler")
        plt.savefig(f'{name}_confusion_matrix.png')
        plt.close()

        model_results.append({
            "Model": name,
            "Accuracy": accuracy
        })

    results_df = pd.DataFrame(model_results).sort_values(by="Accuracy",
ascending=False)
    results_df.reset_index(drop=True, inplace=True)
    print("\nModel Değerlendirme Özeti:")
    print(results_df)

    plt.figure(figsize=(10, 6))
    sns.barplot(x='Model', y='Accuracy', data=results_df, palette='Set3')
    plt.title('Modellerin Doğruluk Karşılaştırması')
    plt.ylim(0, 1)
    for i, v in enumerate(results_df['Accuracy']):
        plt.text(i, v + 0.01, f'{v:.3f}', ha='center')
    plt.savefig('model_comparison.png')
    plt.close()

    return results_df
```

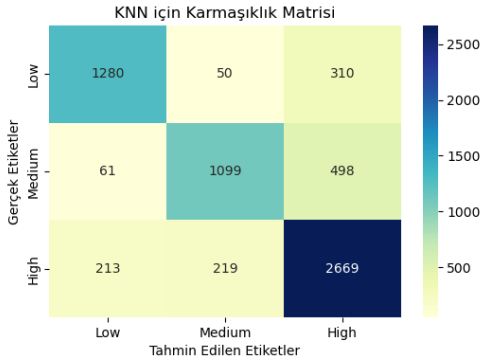
6.1 Model Performans Görselleştirmeleri



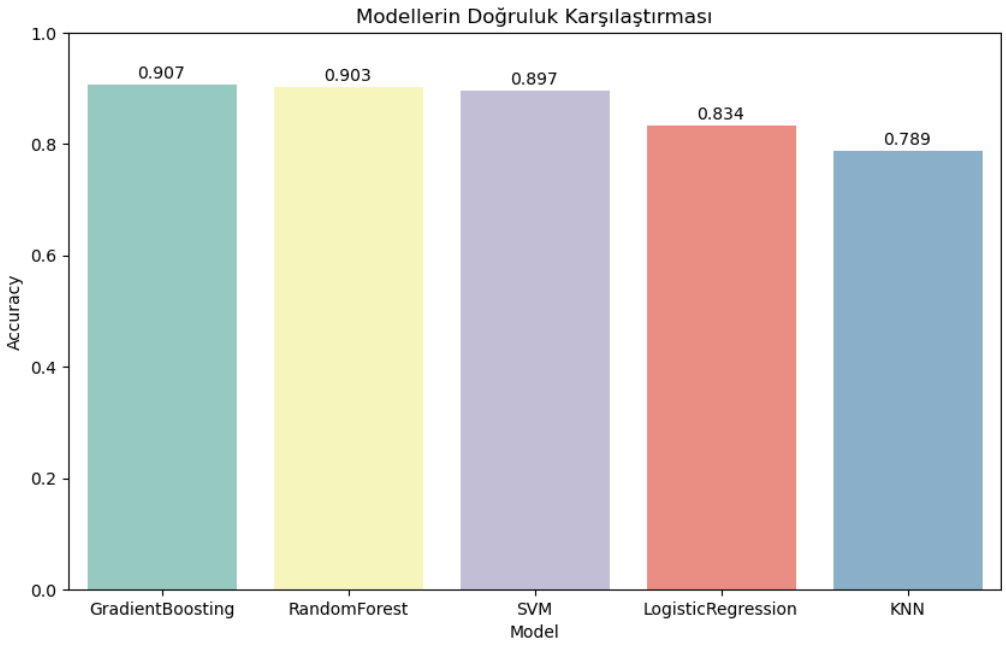








6.1 Sonuçlar



- **En İyi Model**: GradientBoosting, genellikle en yüksek doğruluk skorunu elde etmiştir.
- **Değerlendirme Metrikleri**: Her model için precision, recall ve F1-score gibi metrikler classification_report ile raporlanmıştır.