LIFE EXPECTANCY PREDICTION USING MACHINE LEARNING

ARİF TUNÇER - 2410205529

YAREN YAZAR - 2210213033

SUDENUR ATEŞ - 2210213040

ESRA TAVŞAN - 2110213057

YUSUF CAN DALCI - 2210213032

GROUP ORGANIZATION:

YAREN YAZAR	Project definition, finding the data set, preparation of documentation
ESRA TAVŞAN	Visualization, cleaning and structuring of the dataset, making it suitable for training models
YUSUF CAN DALCI	Training of models and model comparisons, selection of appropriate model
SUDENUR ATEŞ	Calculation of metrics of the most suitable selected model, development of the model
ARİF TUNÇER	Deploying the model on the server with Flask API, preparing a mobile interface, displaying the prediction as a result of the request

Project Description:

Objective:

The aim of this project is to predict the average life expectancy of countries based on various socio-economic and health-related statistical indicators. The goal is to improve prediction accuracy using machine learning techniques and determine the most suitable algorithm for this task.

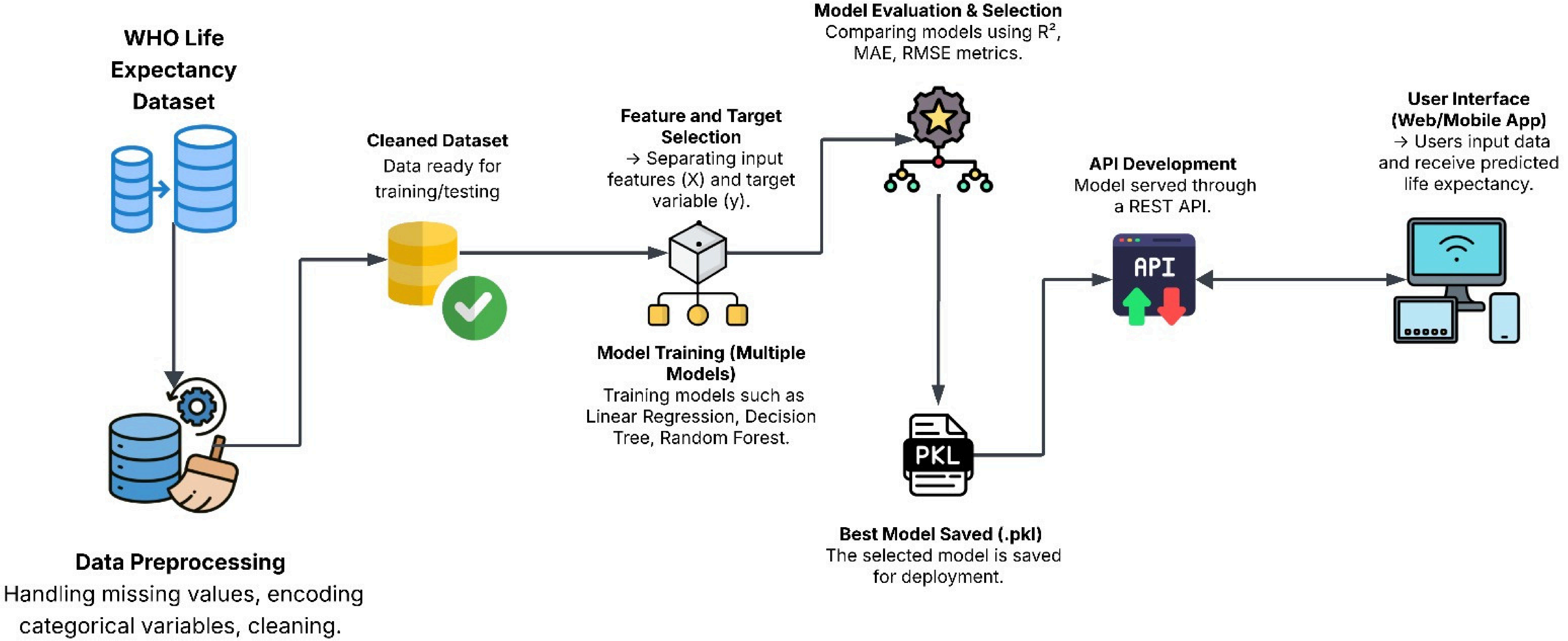
Scope:

The project utilizes the "Life Expectancy (WHO)" dataset provided by the World Health Organization (WHO). The dataset includes a wide range of features such as health expenditure, infectious diseases, and economic indicators for multiple countries.
The scope covers data analysis, preprocessing, regression modeling, creating an API using Flask, and displaying results through an Android interface.

Problem Definition:

Life expectancy is influenced by many factors and varies significantly across countries. This project aims to predict the average life expectancy using a data-driven approach, taking into account the health and economic conditions of each country.
Furthermore, different machine learning models are compared to identify the one that provides the best predictive performance.

PROJECT FLOW CHART:



1.DATASET AND DATA PREPROCESSING:

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

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from google.colab import files
```

```
[ ]  uploaded = files.upload()
```

```
[ ]  df = pd.read_csv("Life Expectancy Data.csv")
```

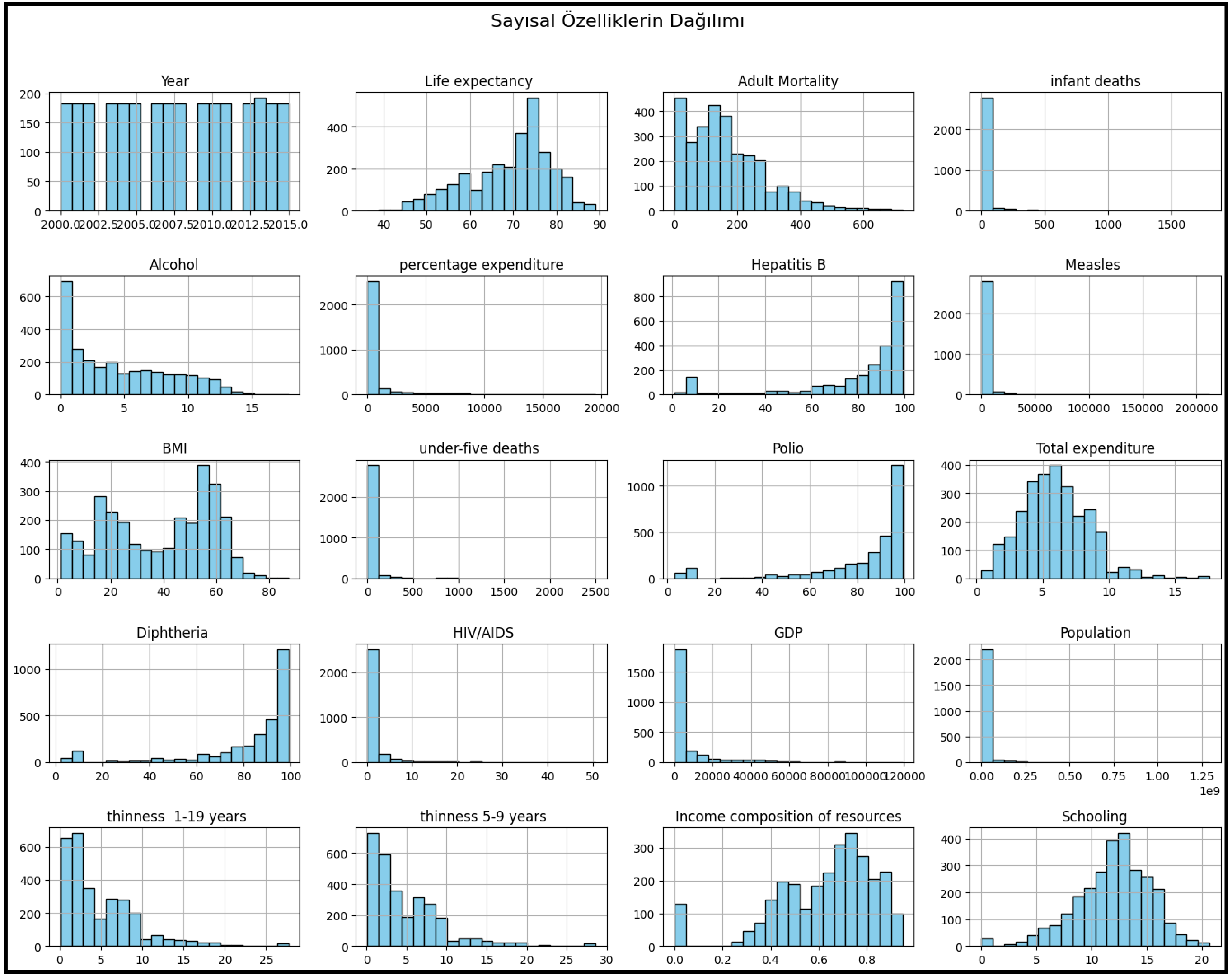
```
df.head()
```

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population	thinness 1-19 years	thinness 5-9 years	Income composition of resources	Schooling
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	8.16	65.0	0.1	584.259210	33736494.0	17.2	17.3	0.479	10.1
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	8.18	62.0	0.1	612.696514	327582.0	17.5	17.5	0.476	10.0
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	8.13	64.0	0.1	631.744976	31731688.0	17.7	17.7	0.470	9.9
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	...	67.0	8.52	67.0	0.1	669.959000	3696958.0	17.9	18.0	0.463	9.8
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	7.87	68.0	0.1	63.537231	2978599.0	18.2	18.2	0.454	9.5

5 rows × 22 columns

```python
print("Veri setindeki tüm sütunlar:")
for i, col in enumerate(df.columns):
    print(f"{i+1}. {col}")
```

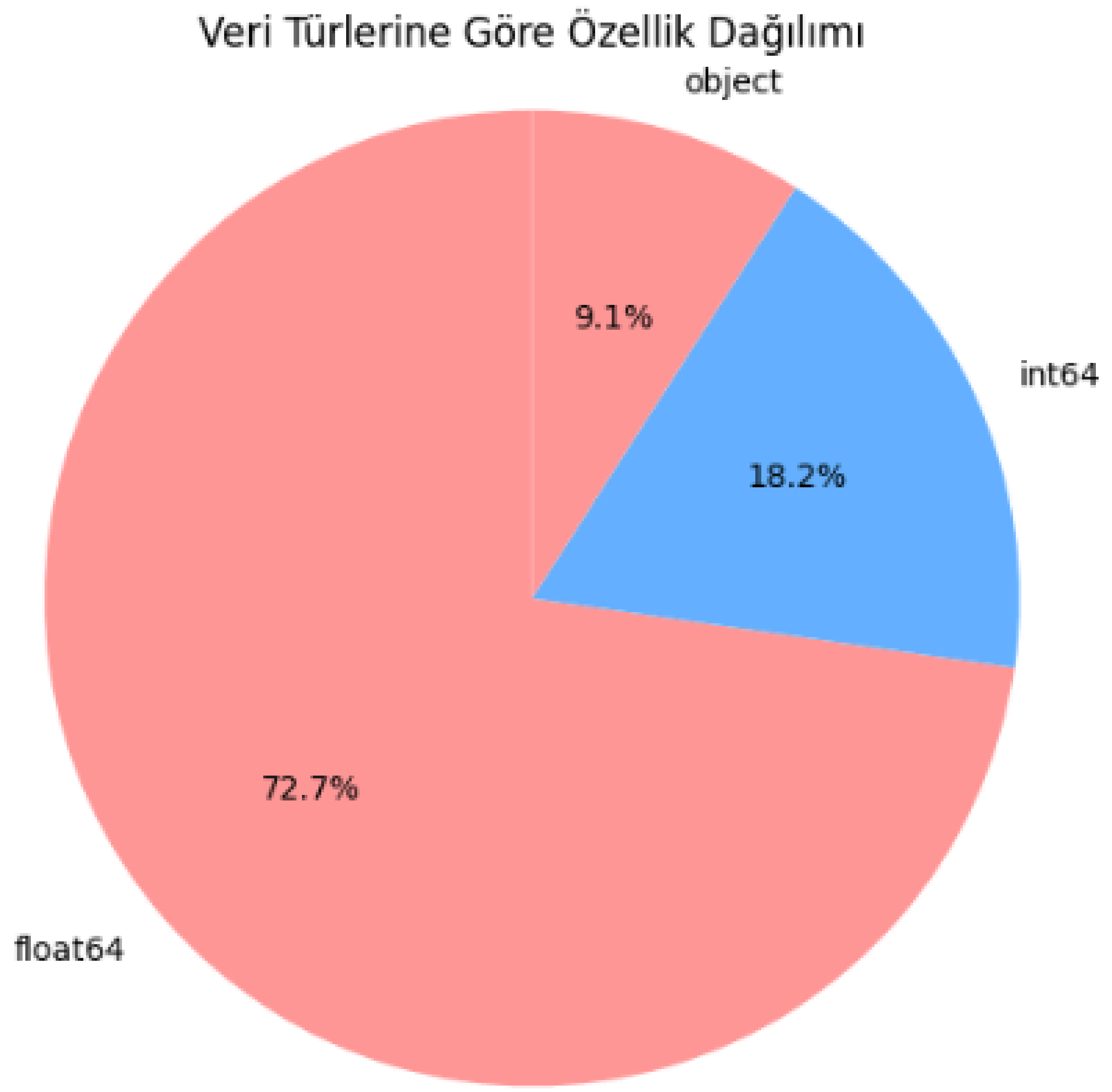
```
Veri setindeki tüm sütunlar:
1. Country
2. Year
3. Status
4. Life expectancy
5. Adult Mortality
6. infant deaths
7. Alcohol
8. percentage expenditure
9. Hepatitis B
10. Measles
11.  BMI
12. under-five deaths
13. Polio
14. Total expenditure
15. Diphtheria
16.  HIV/AIDS
17. GDP
18. Population
19.  thinness  1-19 years
20.  thinness 5-9 years
21. Income composition of resources
22. Schooling
```


Sayısal Özelliklerin Dağılımı
Year
Life expectancy
Adult Mortality
infant deaths
Alcohol
percentage expenditure
Hepatitis B
Measles
BMI
under-five deaths
Polio
Total expenditure
Diphtheria
HIV/AIDS
GDP
Population
1e9
thinness 1-19 years
thinness 5-9 years
Income composition of resources
Schooling

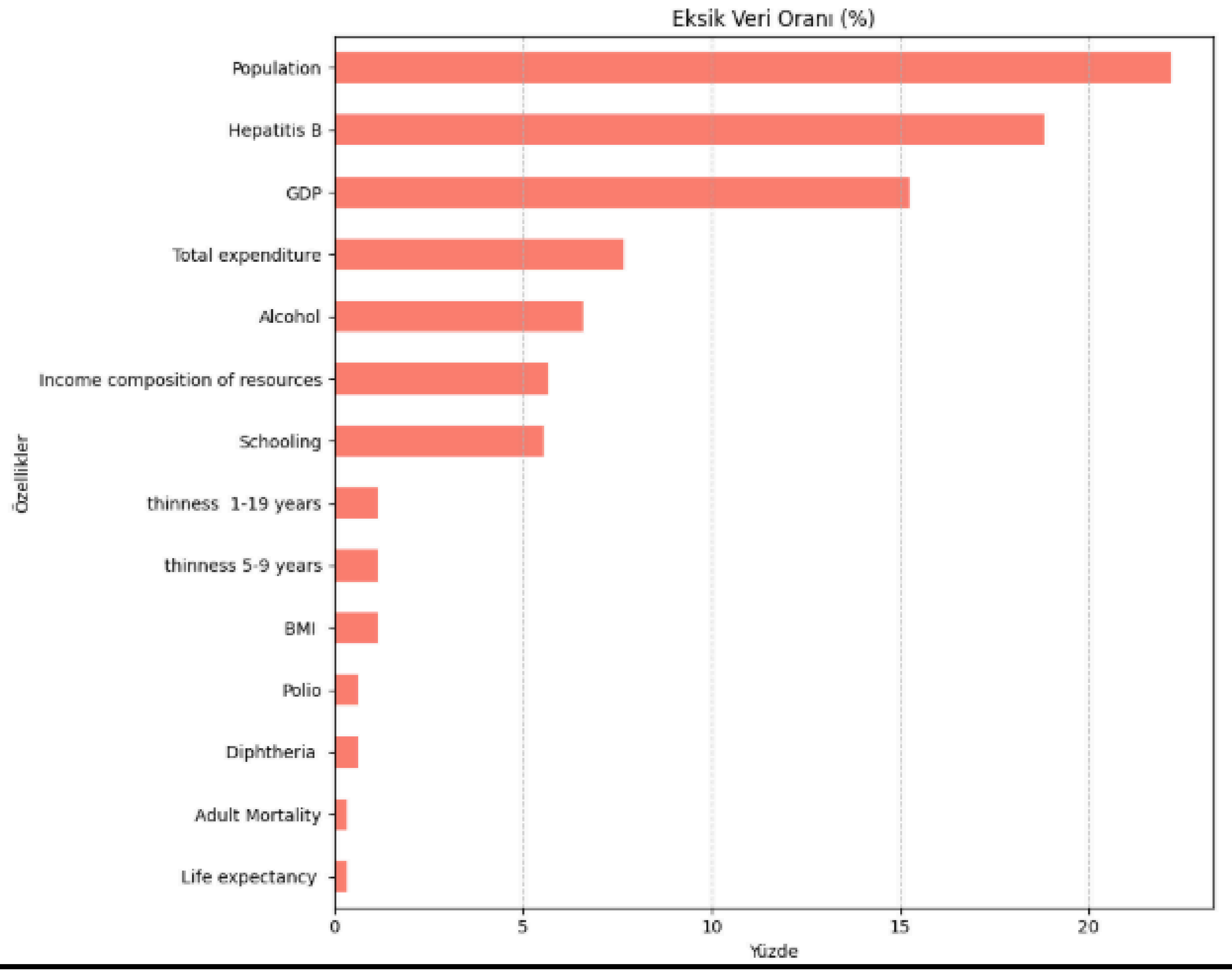
```
categorical_cols = df.select_dtypes(include=['object']).columns.tolist()
print("Kategorik sütunlar:", categorical_cols)
```

```
Kategorik sütunlar: ['Country', 'Status']
```

```
type_counts = df.dtypes.value_counts()

plt.figure(figsize=(6,6))
plt.pie(type_counts, labels=type_counts.index.astype(str), autopct='%1.1f%%', startangle=90, colors=['#ff9999','#66b3ff'])
plt.title("Veri Türlerine Göre Özellik Dağılımı")
plt.axis('equal')
plt.show()
```



```
missing = df.isnull().mean() * 100
missing = missing[missing > 0].sort_values()

plt.figure(figsize=(10, 8))
missing.plot(kind='barh', color='salmon')
plt.title("Eksik Veri Oranı (%)")
plt.xlabel("Yüzde")
plt.ylabel("Özellikler")
plt.grid(True, axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



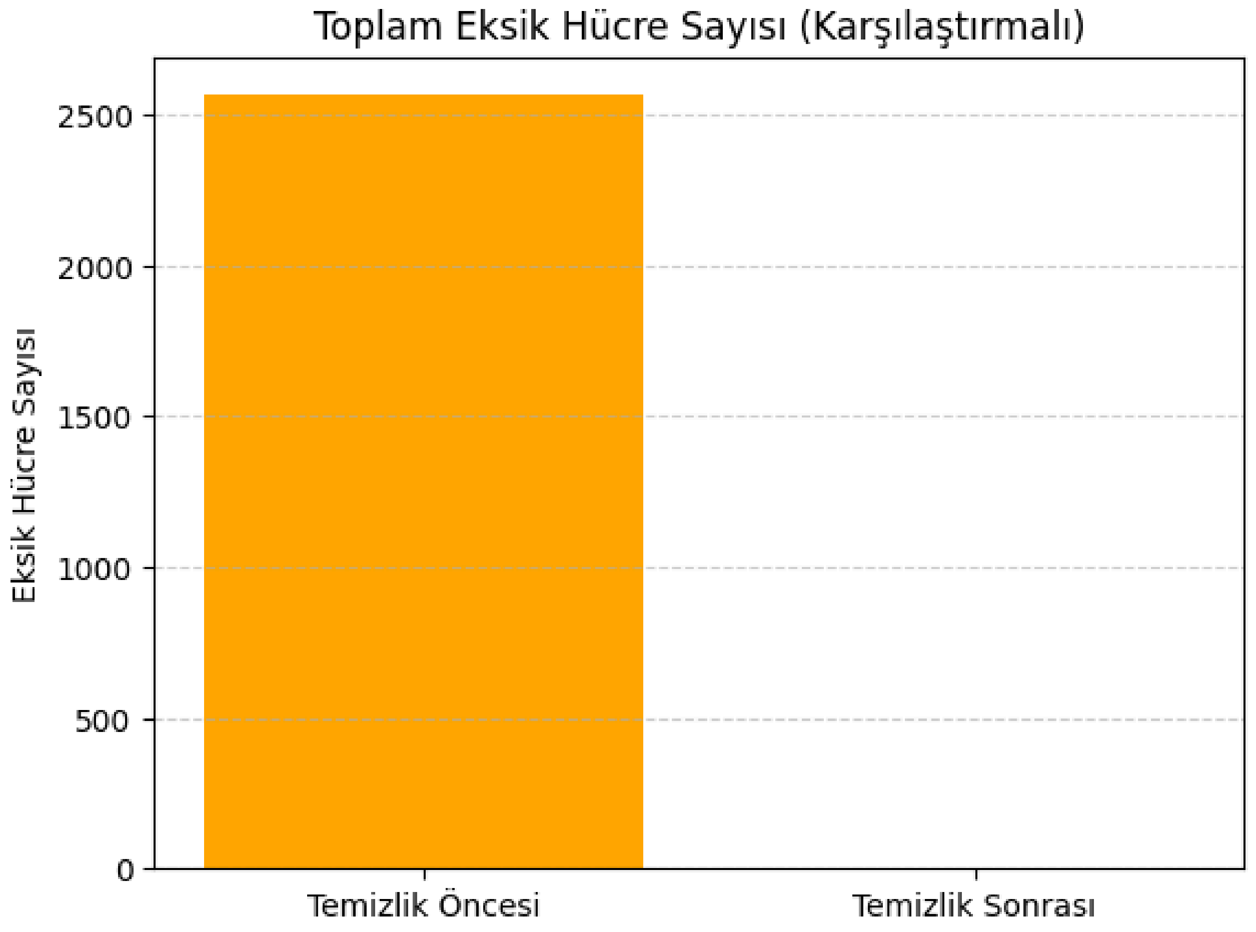
2.CLEANED DATASET:

```
df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
```

```
raw_df = pd.read_csv("Life Expectancy Data.csv")
```

```
df = df.fillna(df.mean(numeric_only=True))
```

```
total_missing_raw = raw_df.isnull().sum().sum()
total_missing_clean = df.isnull().sum().sum()

plt.bar(["Temizlik Öncesi", "Temizlik Sonrası"], [total_missing_raw, total_missing_clean], color=["orange", "green"])
plt.ylabel("Eksik Hücre Sayısı")
plt.title("Toplam Eksik Hücre Sayısı (Karşılaştırmalı)")
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.show()
```



3.MODEL TRAINING (MULTIPLE MODELS) / MODEL EVALUATION & SELECTION:

LINEAR REGRESSION:

```
[ ]  y = df["Life expectancy "]
     X = df.drop(columns=["Life expectancy "])

     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(
         X, y,
         test_size=0.2,
         random_state=42
     )
```

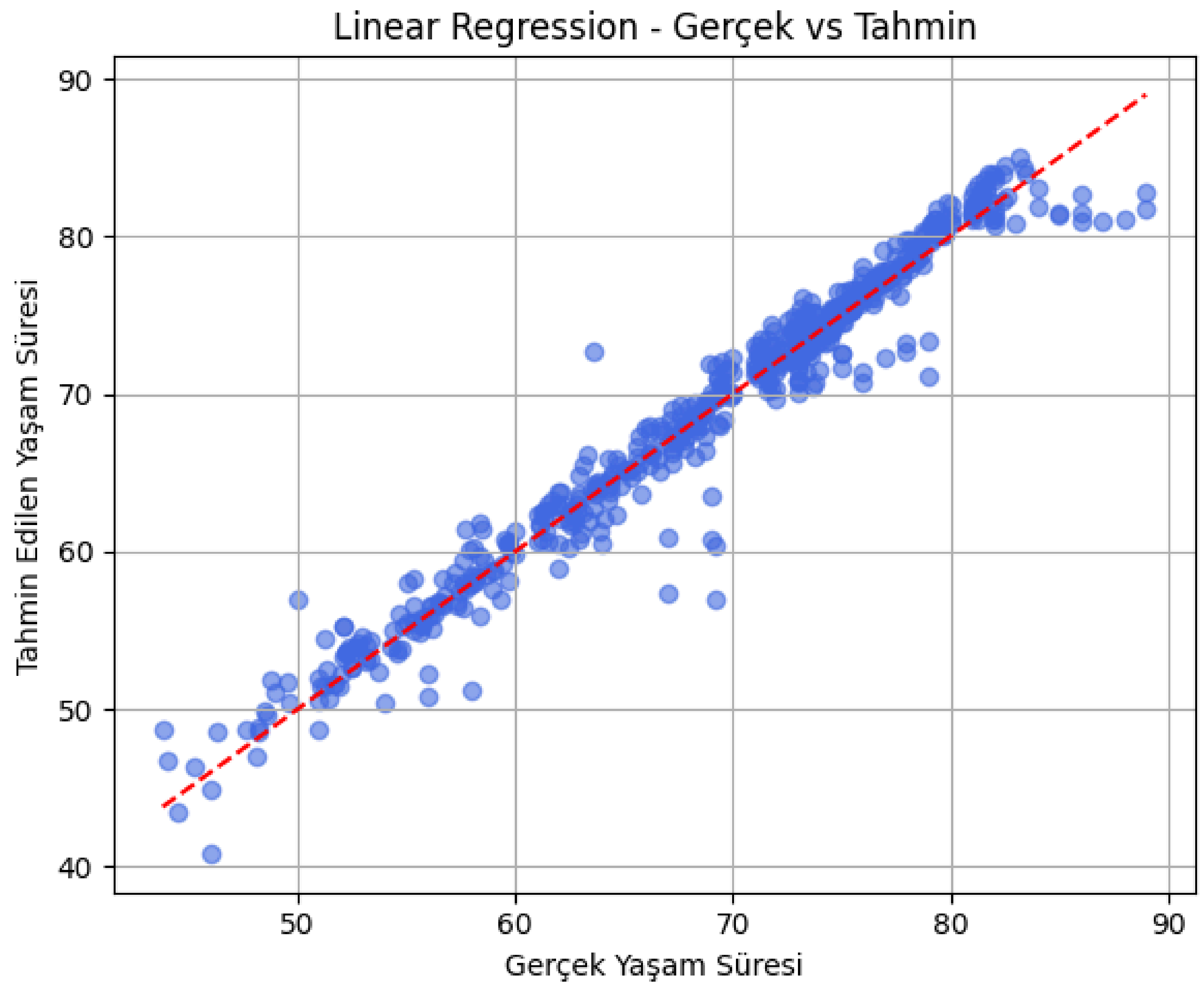
```
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
```

```
▾ LinearRegression ℹ ?
LinearRegression()
```

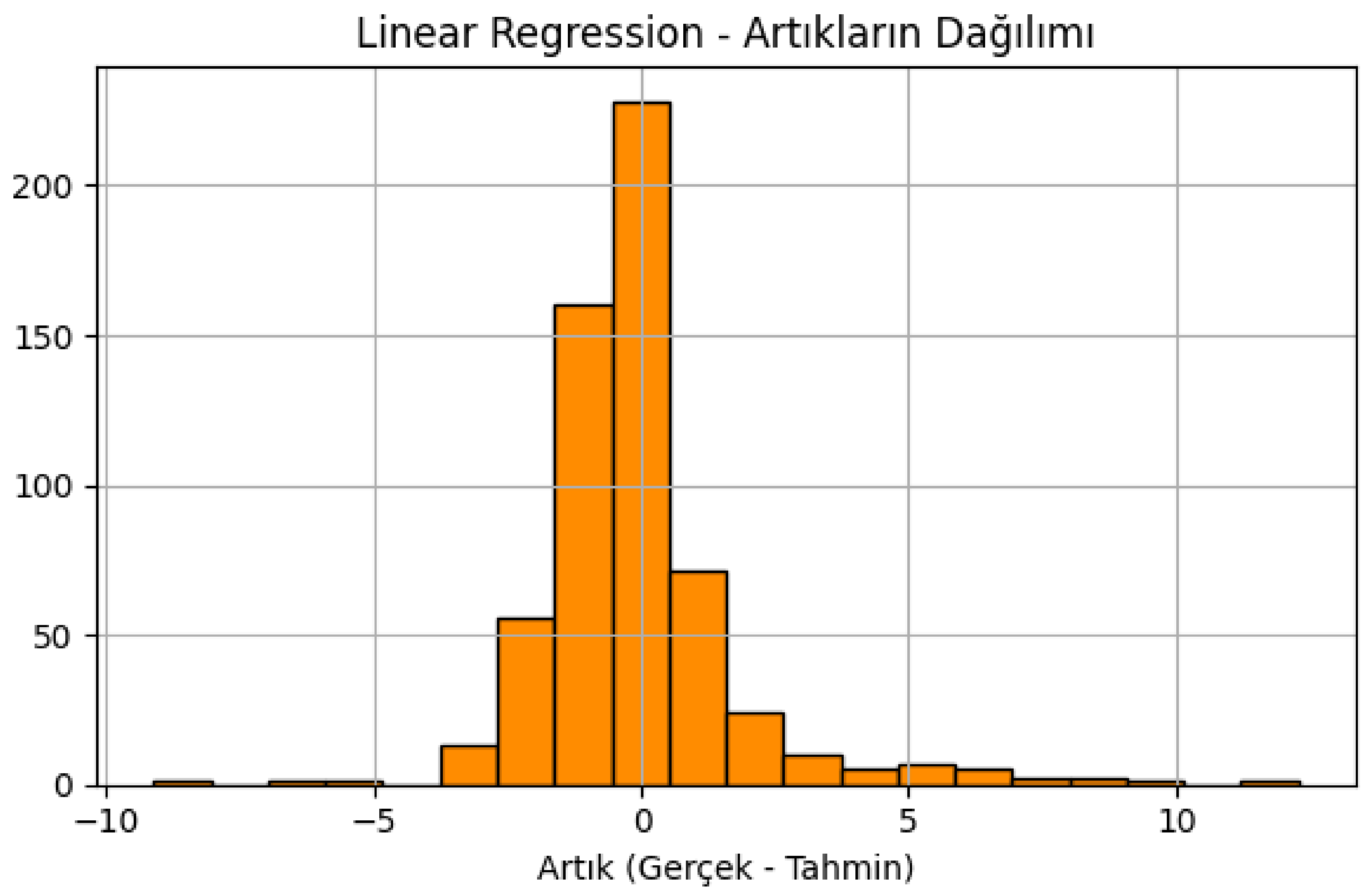
```
y_pred_lr = lr_model.predict(X_test)

mae_lr = mean_absolute_error(y_test, y_pred_lr)
rmse_lr = np.sqrt(mean_squared_error(y_test, y_pred_lr))
r2_lr = r2_score(y_test, y_pred_lr)

print(" 🔹 Linear Regression Sonuçları:")
print(f"MAE  : {mae_lr:.2f}")
print(f"RMSE : {rmse_lr:.2f}")
print(f"R²   : {r2_lr:.2f}")
```

```
🔹 Linear Regression Sonuçları:
MAE  : 1.19
RMSE : 1.85
R²   : 0.96
```

```python
plt.figure(figsize=(6,5))
plt.scatter(y_test, y_pred_lr, alpha=0.6, color='royalblue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel("Gerçek Yaşam Süresi")
plt.ylabel("Tahmin Edilen Yaşam Süresi")
plt.title("Linear Regression - Gerçek vs Tahmin")
plt.grid(True)
plt.tight_layout()
plt.show()
```




```python
residuals_lr = y_test - y_pred_lr
plt.figure(figsize=(6,4))
plt.hist(residuals_lr, bins=20, color='darkorange', edgecolor='black')
plt.title("Linear Regression - Artıkların Dağılımı")
plt.xlabel("Artık (Gerçek - Tahmin)")
plt.grid(True)
plt.tight_layout()
plt.show()
```



What Can We Conclude?

#Linear regression performs well for average predictions.

#However, for some points, a linear relationship might be insufficient.

In this case:

#Trying non-linear models (such as Decision Tree, Random Forest) would be reasonable.

#Additionally, observing and filtering out outliers may improve the model's quality.

DECISION TREE:

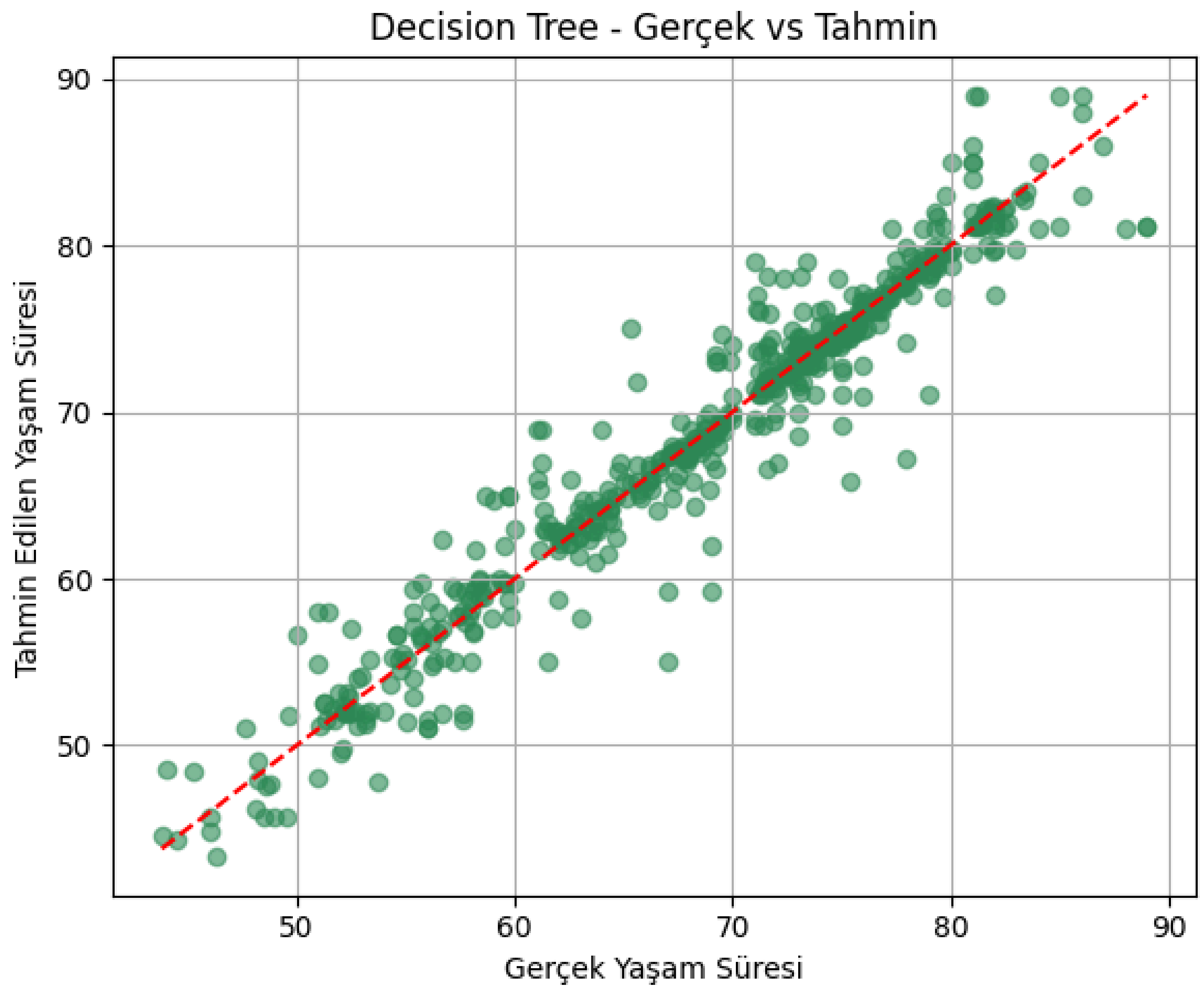
```
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)
```

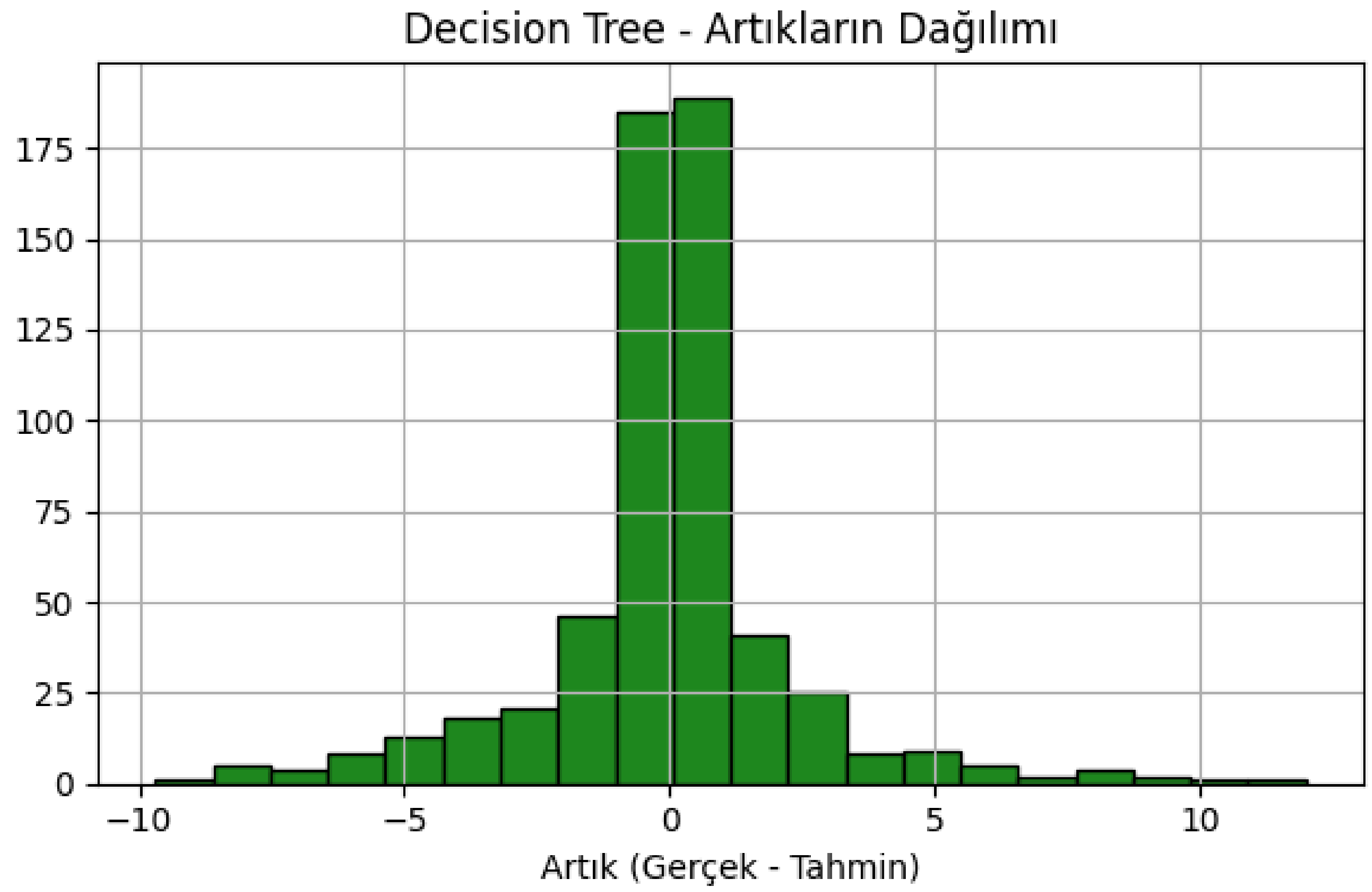
```
y_pred_dt = dt_model.predict(X_test)

mae_dt = mean_absolute_error(y_test, y_pred_dt)
rmse_dt = np.sqrt(mean_squared_error(y_test, y_pred_dt))
r2_dt = r2_score(y_test, y_pred_dt)

print("🌳 Decision Tree Regressor Sonuçları:")
print(f"MAE  : {mae_dt:.2f}")
print(f"RMSE : {rmse_dt:.2f}")
print(f"R²   : {r2_dt:.2f}")
```

```
🌳 Decision Tree Regressor Sonuçları:
MAE  : 1.46
RMSE : 2.40
R²   : 0.93
```

```python
plt.figure(figsize=(6,5))
plt.scatter(y_test, y_pred_dt, alpha=0.6, color='seagreen')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel("Gerçek Yaşam Süresi")
plt.ylabel("Tahmin Edilen Yaşam Süresi")
plt.title("Decision Tree - Gerçek vs Tahmin")
plt.grid(True)
plt.tight_layout()
plt.show()
```




```python
residuals_dt = y_test - y_pred_dt
plt.figure(figsize=(6,4))
plt.hist(residuals_dt, bins=20, color='forestgreen', edgecolor='black')
plt.title("Decision Tree - Artıkların Dağılımı")
plt.xlabel("Artık (Gerçek - Tahmin)")
plt.grid(True)
plt.tight_layout()
plt.show()
```



RANDOM FOREST REGRESSOR:

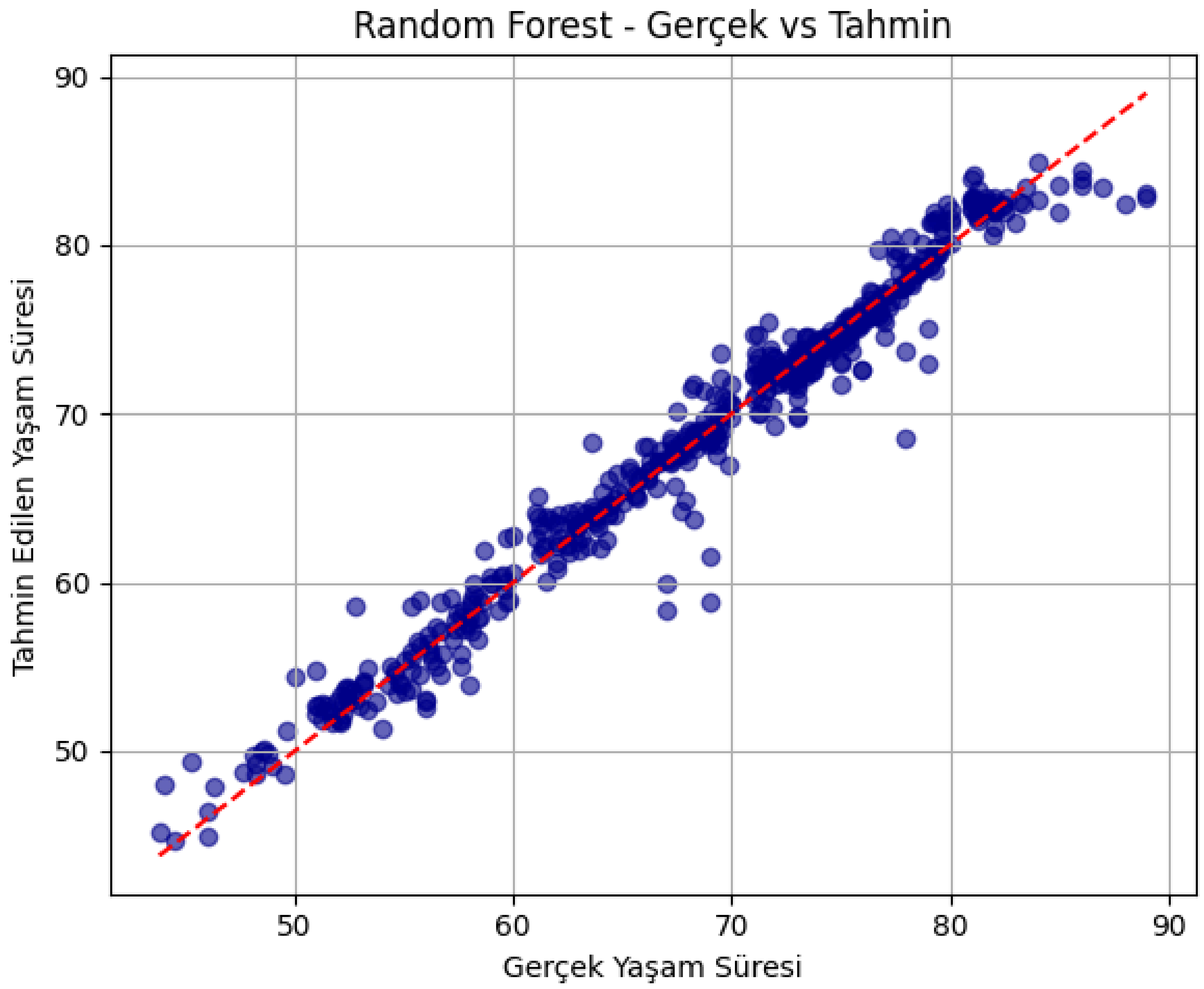
```
[ ] rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
    rf_model.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

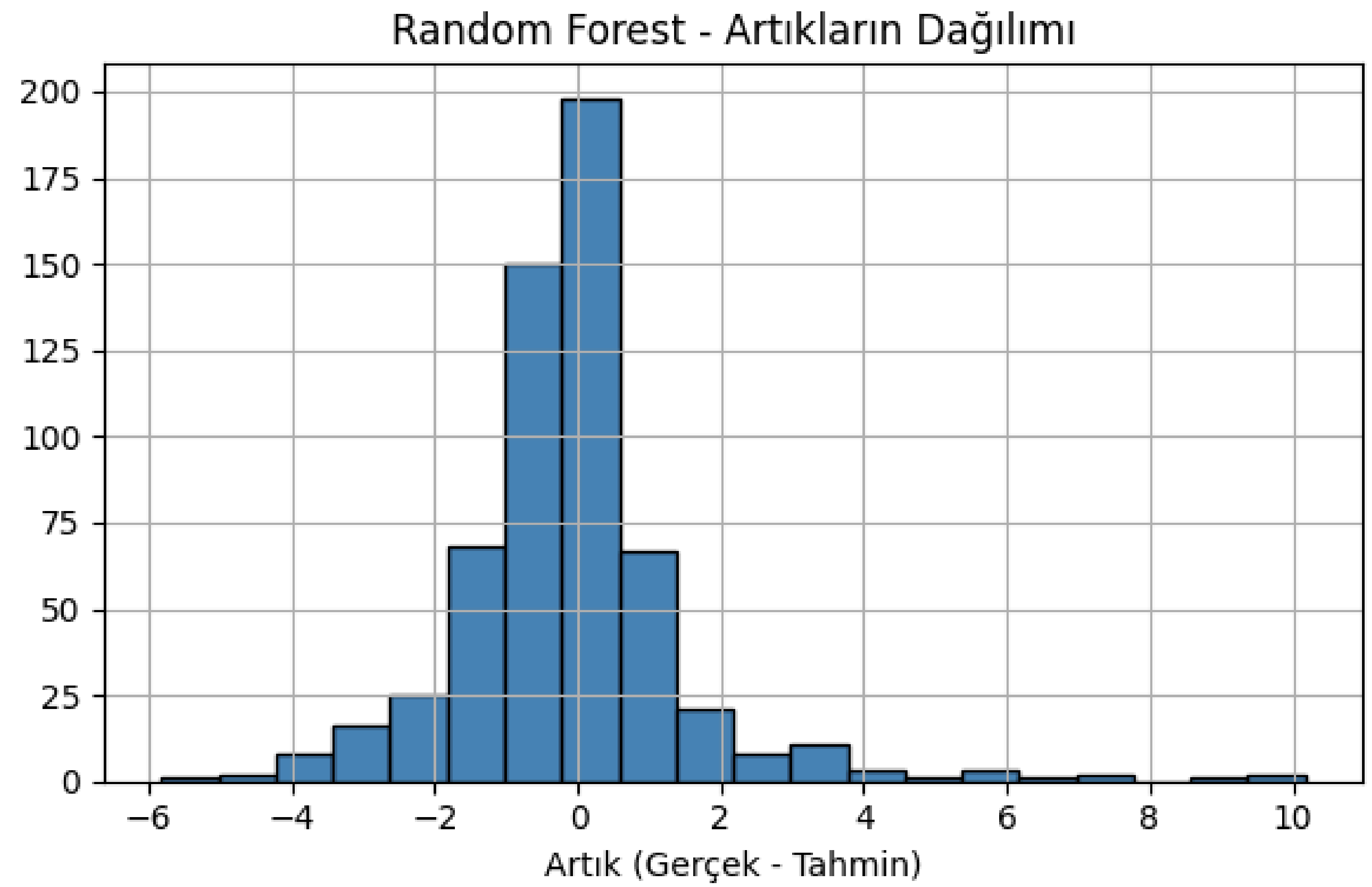
```
[ ] y_pred_rf = rf_model.predict(X_test)

    mae_rf = mean_absolute_error(y_test, y_pred_rf)
    rmse_rf = np.sqrt(mean_squared_error(y_test, y_pred_rf))
    r2_rf = r2_score(y_test, y_pred_rf)

    print("🌲 Random Forest Regressor Sonuçları:")
    print(f"MAE  : {mae_rf:.2f}")
    print(f"RMSE : {rmse_rf:.2f}")
    print(f"R²   : {r2_rf:.2f}")
```

```
🌲 Random Forest Regressor Sonuçları:
MAE  : 1.03
RMSE : 1.62
R²   : 0.97
```

```
[ ] plt.figure(figsize=(6,5))
    plt.scatter(y_test, y_pred_rf, alpha=0.6, color='darkblue')
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
    plt.xlabel("Gerçek Yaşam Süresi")
    plt.ylabel("Tahmin Edilen Yaşam Süresi")
    plt.title("Random Forest - Gerçek vs Tahmin")
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

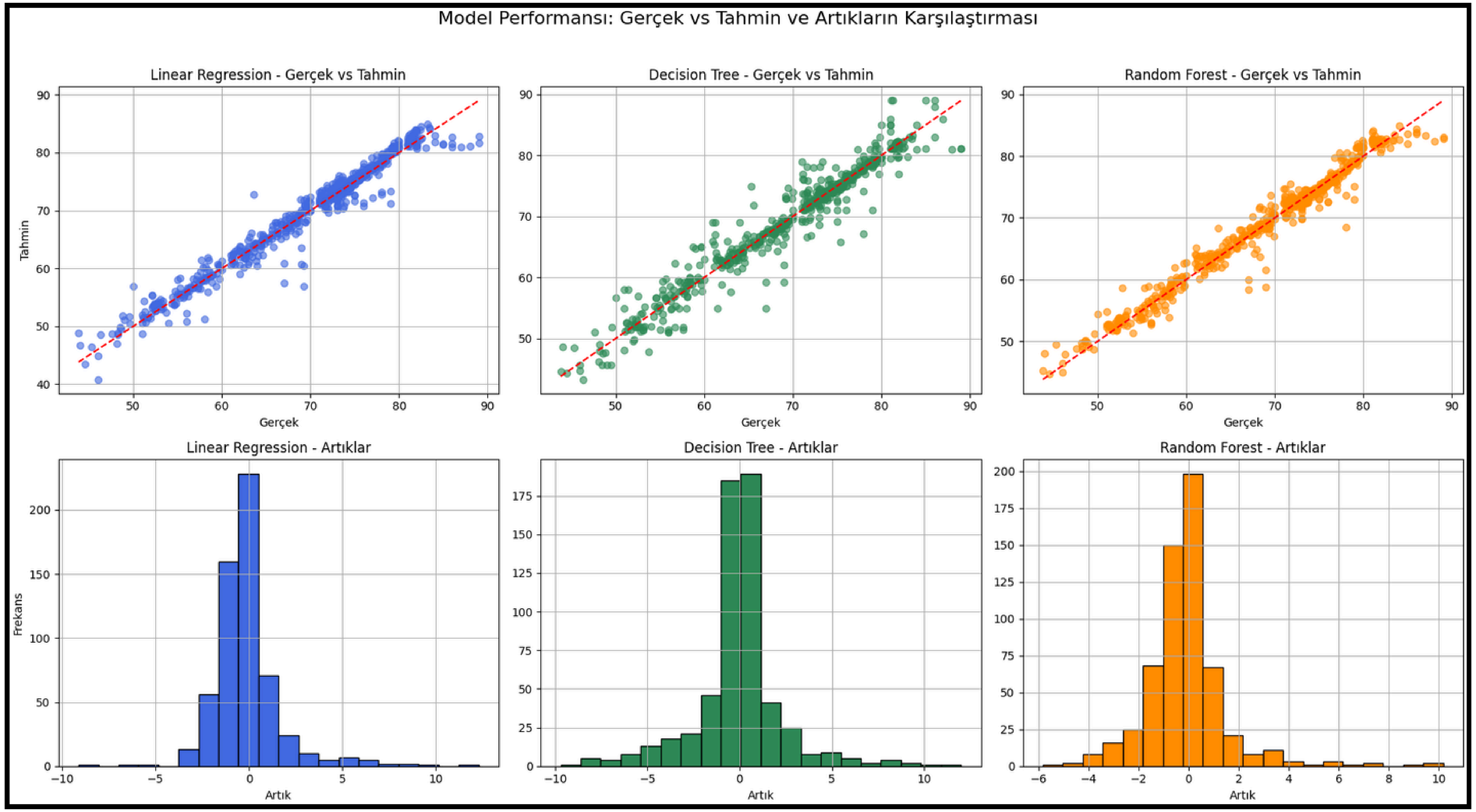



```
[ ] residuals_rf = y_test - y_pred_rf
    plt.figure(figsize=(6,4))
    plt.hist(residuals_rf, bins=20, color='steelblue', edgecolor='black')
    plt.title("Random Forest - Artıkların Dağılımı")
    plt.xlabel("Artık (Gerçek - Tahmin)")
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```



📊 Üç Modelin Performans Karşılaştırması

	Model	MAE	RMSE	R²
0	Linear Regression	1.185378	1.853586	0.960342
1	Decision Tree	1.458163	2.402366	0.933383
2	Random Forest	1.032129	1.620102	0.969704



BEST MODEL:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

y_pred_train = rf_model.predict(X_train)
y_pred_test = rf_model.predict(X_test)

r2_train = r2_score(y_train, y_pred_train)
mae_train = mean_absolute_error(y_train, y_pred_train)
rmse_train = np.sqrt(mean_squared_error(y_train, y_pred_train))

r2_test = r2_score(y_test, y_pred_test)
mae_test = mean_absolute_error(y_test, y_pred_test)
rmse_test = np.sqrt(mean_squared_error(y_test, y_pred_test))

metrics_df = pd.DataFrame({
    "Veri Seti": ["Eğitim", "Test"],
    "R²": [r2_train, r2_test],
    "MAE": [mae_train, mae_test],
    "RMSE": [rmse_train, rmse_test]
})
print("📊 Overfitting/Underfitting Karşılaştırması\n")
display(metrics_df)

print("\n🔍 Yorumlama:")

r2_gap = r2_train - r2_test

if r2_gap > 0.1 and r2_test < 0.9:
    print("⚠️ Overfitting tespit edildi: Eğitim R² çok yüksek, test R² düşük.")
elif r2_train < 0.8 and r2_test < 0.8:
    print("📉 Underfitting tespit edildi: Model hem eğitim hem test verisinde başarısız.")
elif abs(r2_gap) <= 0.1 and r2_test >= 0.9:
    print("✅ Model genelleme başarısı yüksek. Overfitting / underfitting görünmüyor.")
else:
    print("ℹ️ Model dengeli olabilir, ancak skorlar incelenmeli.")
```

📊 Overfitting/Underfitting Karşılaştırması

	Veri Seti	R²	MAE	RMSE
0	Eğitim	0.994680	0.423849	0.696853
1	Test	0.969704	1.032129	1.620102

🔍 Yorumlama:
✅ Model genelleme başarısı yüksek. Overfitting / underfitting görünmüyor.

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer

cv_scores_r2 = cross_val_score(rf_model, X, y, cv=5, scoring='r2')

mae_scorer = make_scorer(mean_absolute_error, greater_is_better=False)
cv_scores_mae = cross_val_score(rf_model, X, y, cv=5, scoring=mae_scorer)

print("📊 Cross-Validation Skorları (5-Fold):")
print("R² Skorları:", np.round(cv_scores_r2, 4))
print("Ortalama R²:", np.mean(cv_scores_r2).round(4))
print("\nMAE Skorları:", np.round(-cv_scores_mae, 4))
print("Ortalama MAE:", (-np.mean(cv_scores_mae)).round(4))
```

```
📊 Cross-Validation Skorları (5-Fold):
R² Skorları: [0.9378 0.8779 0.9161 0.9088 0.9081]
Ortalama R²: 0.9097

MAE Skorları: [1.9331 2.2983 1.8746 1.8631 2.0066]
Ortalama MAE: 1.9951
```

```
mean_r2 = np.mean(cv_scores_r2)
mean_mae = -np.mean(cv_scores_mae)

comparison_df = pd.DataFrame({
    "Metrik": ["R²", "MAE"],
    "Cross-Validation (Ortalama)": [mean_r2, mean_mae],
    "Test Skoru": [r2_test, mae_test]
})

print("📊 Cross-Validation vs Test Skoru Karşılaştırması")
print(comparison_df)
```

```
📊 Cross-Validation vs Test Skoru Karşılaştırması
  Metrik  Cross-Validation (Ortalama)  Test Skoru
0     R²                     0.909728    0.969704
1    MAE                     1.995142    1.032129
```

```
r2_gap_percent = abs(r2_test - mean_r2) / r2_test * 100

print(f"📐 Test ve Cross-Validation R² farkı: {r2_gap_percent:.2f}%")

print("🔍 Yorumlama:")
if r2_gap_percent <= 5:
    print("✅ Model genelleme konusunda çok başarılı. Ezberleme (overfitting) yapmıyor.")
elif r2_gap_percent <= 10:
    print("🟡 Model genellemesi kabul edilebilir seviyede. Küçük fark doğal.")
else:
    print("❗ Model yüksek ihtimalle overfitting yapıyor. Farklı veri setlerinde performansı düşebilir.")
```

```
📐 Test ve Cross-Validation R² farkı: 6.18%
🔍 Yorumlama:
🟡 Model genellemesi kabul edilebilir seviyede. Küçük fark doğal.
```

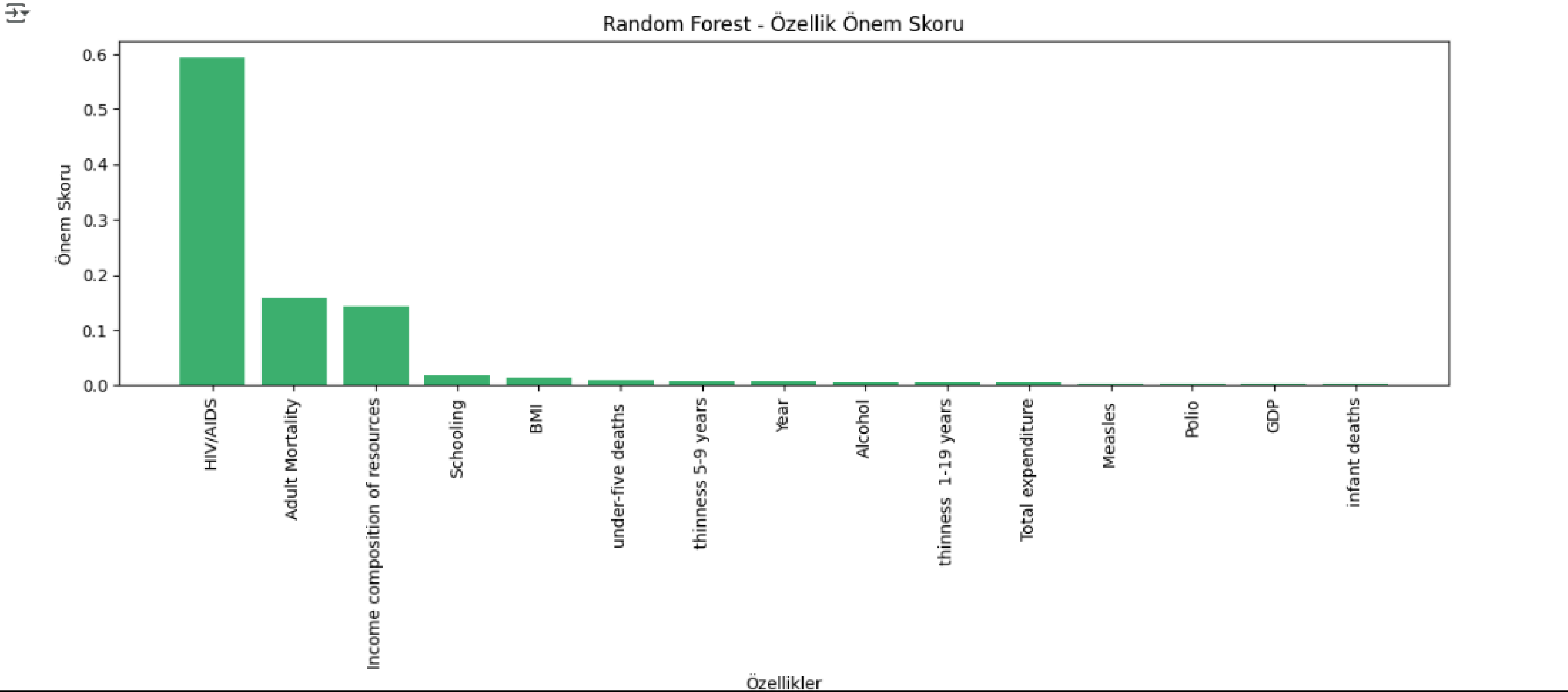
#The most suitable model was selected as the Random Forest model.

#Generalization was evaluated using Cross-Validation, and a certain level of overfitting was observed.

#To reduce the risk of overfitting, steps were taken to improve the Random Forest model.

#Let's proceed with the Random Forest model optimization steps.

```python
feature_names = X.columns
importances = rf_model.feature_importances_
indices = np.argsort(importances)[::-1]

plt.figure(figsize=(12, 6))
plt.title("Random Forest - Özellik Önem Skoru")
plt.bar(range(15), importances[indices[:15]], align="center", color="mediumseagreen")
plt.xticks(range(15), feature_names[indices[:15]], rotation=90)
plt.xlabel("Özellikler")
plt.ylabel("Önem Skoru")
plt.tight_layout()
plt.show()

top_features_df = pd.DataFrame({
    "Özellik": feature_names[indices],
    "Önem Skoru": importances[indices]
}).head(10)

top_features_df
```



	Özellik	Önem Skoru
0	HIV/AIDS	0.593354
1	Adult Mortality	0.157995
2	Income composition of resources	0.142379
3	Schooling	0.017789
4	BMI	0.013322
5	under-five deaths	0.010461
6	thinness 5-9 years	0.007848
7	Year	0.006792
8	Alcohol	0.006108
9	thinness 1-19 years	0.005260

```python
from sklearn.model_selection import GridSearchCV

# Hiperparametre aralığı
param_grid = {
    'n_estimators': [100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'max_features': ['sqrt', 'log2']
}

grid_search = GridSearchCV(
    estimator=RandomForestRegressor(random_state=42),
    param_grid=param_grid,
    cv=5,
    scoring='r2',
    verbose=1,
    n_jobs=-1
)

grid_search.fit(X_train, y_train)

best_rf = grid_search.best_estimator_
y_pred_best = best_rf.predict(X_test)

from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

r2_best = r2_score(y_test, y_pred_best)
mae_best = mean_absolute_error(y_test, y_pred_best)
rmse_best = np.sqrt(mean_squared_error(y_test, y_pred_best))

print("🔧 En iyi hiperparametreler:")
print(grid_search.best_params_)

print("\n✅ Optimize edilmiş modelin test skorları:")
print(f"R²: {r2_best:.3f}")
print(f"MAE: {mae_best:.3f}")
print(f"RMSE: {rmse_best:.3f}")
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
🔧 En iyi hiperparametreler:
{'max_depth': None, 'max_features': 'sqrt', 'min_samples_split': 2, 'n_estimators': 150}

✅ Optimize edilmiş modelin test skorları:
R²: 0.966
MAE: 1.148
RMSE: 1.706
```

```python
import joblib

joblib.dump(best_rf_narrow, 'life_expectancy_rf_model.pkl')
joblib.dump(list(X.columns), 'model_columns.pkl')
```

Gizli çıkışı göster

The best Random Forest model was determined by performing hyperparameter optimization using GridSearchCV.

4.MOBILE INTERFACE, API POST REQUEST AND PREDICT THE LIFE EXPECTANCY TIME:

PYTHON FLASK API

```python
from flask import Flask, request, jsonify
import joblib
import pandas as pd

# Flask uygulamasını başlat
app = Flask(__name__)

# 📌 .pkl dosyasını yükle (aynı klasörde olmalı)
model = joblib.load("random_forest_model.pkl")

# Tahmin endpoint'i
@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Kullanıcıdan gelen JSON verisini al
        data = request.get_json()

        # Tek satırlık veriyi DataFrame'e çevir
        df = pd.DataFrame([data])

        # Tahmin yap
        prediction = model.predict(df)

        # Tahmini JSON formatında döndür
        return jsonify({'prediction': float(prediction[0])})

    except Exception as e:
        return jsonify({'error': str(e)})

# Uygulamayı çalıştır
if __name__ == '__main__':
    app.run(debug=True)
```

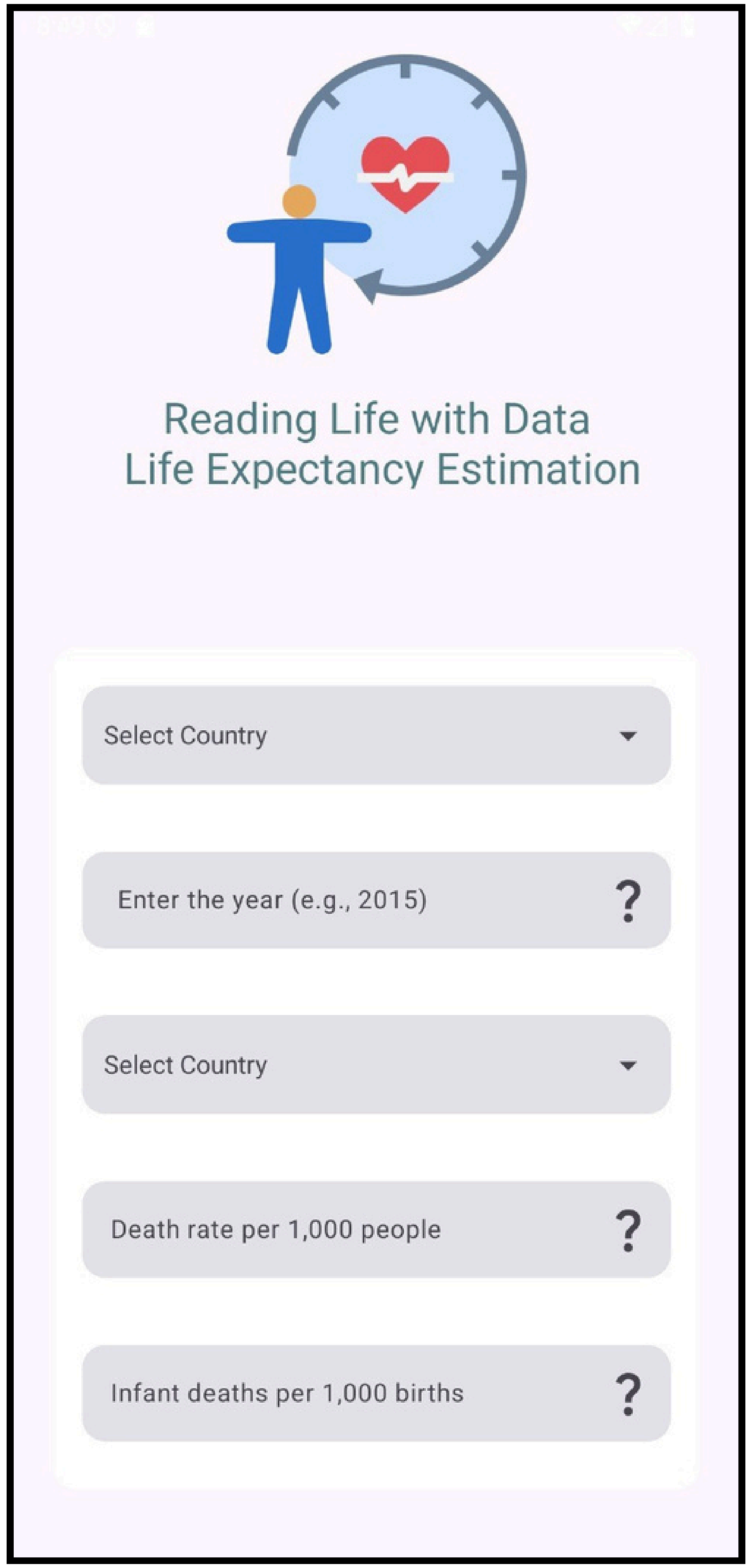
```
Goodbye tunnels, hello Agent Endpoints: https://ngrok.com/r/aep

Session Status                online
Account                       tnqr.arif.61@gmail.com (Plan: Free)
Version                       3.22.1
Region                        Europe (eu)
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://148e-176-219-250-101.ngrok-free.app -> http://localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

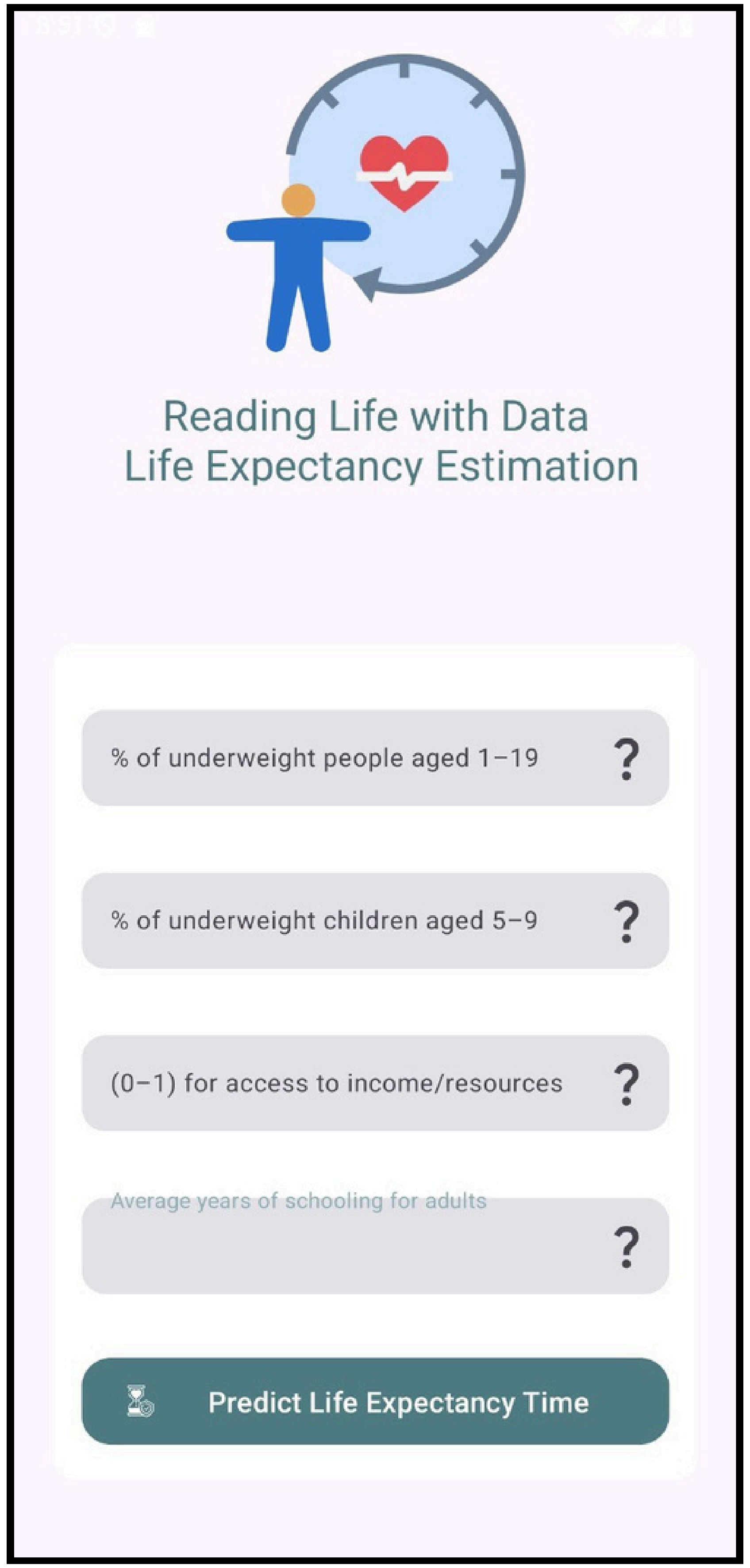
Opening an API to the outside world using ngrok on a server


```
{
    "prediction": 69.9816464516922
}
```

```
{
    "Year": 2025,
    "Adult Mortality": 200,
    "infant deaths": 2,
    "Alcohol": 3.5,
    "percentage expenditure": 3000,
    "Hepatitis B": 85,
    "Measles ": 10,
    " BMI ": 21.4,
    "under-five deaths ": 3,
    "Polio": 90,
    "Total expenditure": 6.3,
    "Diphtheria ": 88,
    " HIV/AIDS": 0.9,
    "GDP": 1500,
    "Population": 2000000,
    " thinness  1-19 years": 3.2,
    " thinness 5-9 years": 1.0,
    "Income composition of resources": 0.65,
    "Schooling": 13.0,
    "Status_Developing": 1
}
```


Using Postman, send a POST request to the ngrok-exposed API endpoint with a JSON body that contains the input features. The server will respond with the predicted life expectancy based on the trained model.





This mobile interface was designed as part of a machine learning project to estimate life expectancy based on key health, education, and economic indicators. Users can input real-world values to receive a prediction generated by a trained Random Forest model.

CONCLUSION:

In this project, life expectancy was predicted using various machine learning algorithms based on the WHO's dataset. Initially, Linear Regression and Decision Tree models were tested, and ultimately, the Random Forest Regressor was selected for its superior performance. The model was evaluated using both train-test splits and cross-validation. The close R^2 scores and low error rates indicated that the model generalizes well and performs reliably. Furthermore, hyperparameter tuning using GridSearchCV improved the model’s accuracy and reduced the risk of overfitting. In conclusion, the Random Forest algorithm proved to be a robust and effective approach for predicting life expectancy and can be a valuable tool for working with socioeconomic health data.

REFERENCES:

#https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who/data

#https://colab.research.google.com/drive/1oQZZvCM7TO-Ev_Op68kXg7Nb3a4UVCr4#scrollTo=k-Qf_22OtsJA