ENERGY MANAGEMENT STRATGIES IN ELECTRIC VEHICLES

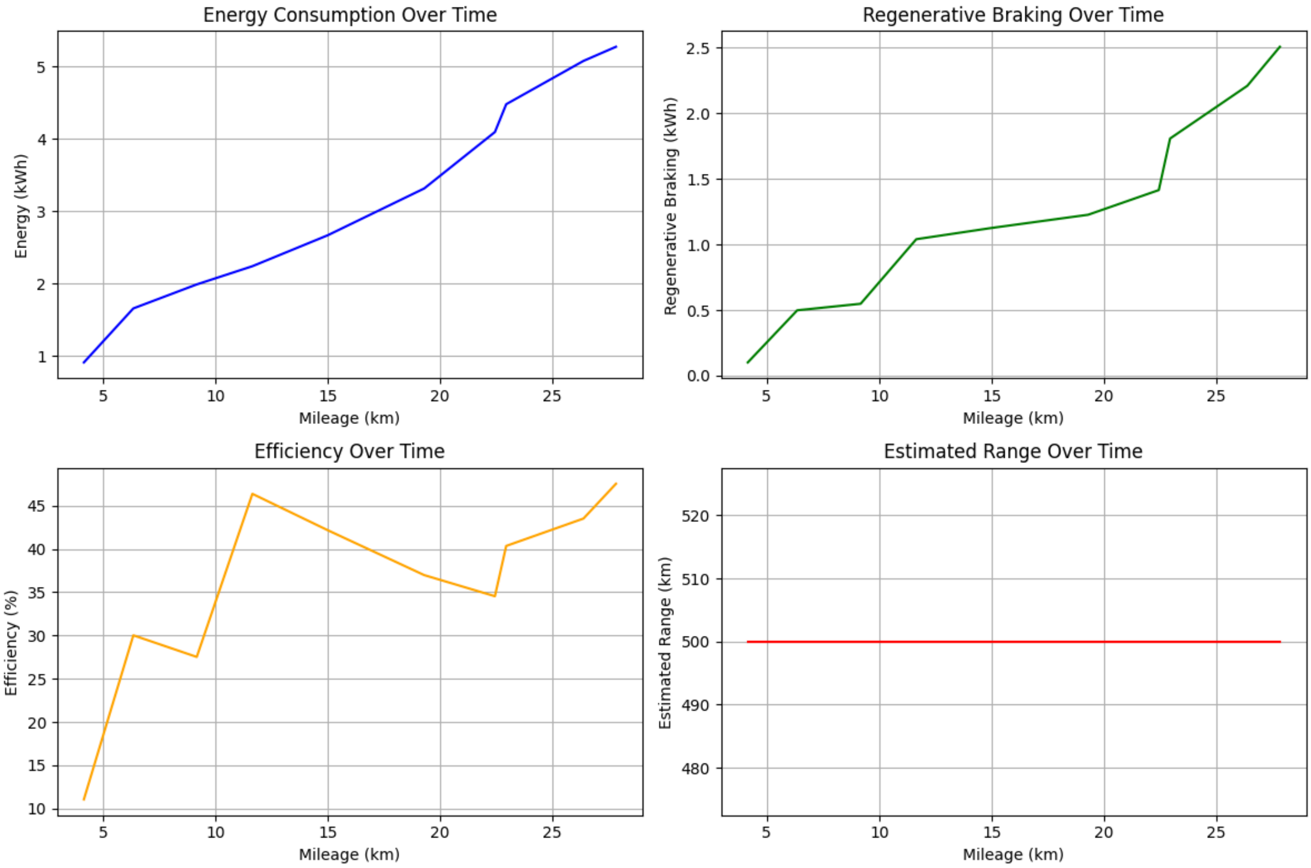
```
import random
import time
import numpy as np
import matplotlib.pyplot as plt
# Basic configuration for EV parameters
class EVparameters:
    def __init__(self):
        self.soc = 100 # state of charge (in percentage)
        self.total_capacity =80 # battery capacity in kwh
        self.current_speed = 0 #speed in km/h
        self.power_consumption = 0 # power consumption in kW
        self.regen_efficiency = 0.7 # efficiency of regenarative braking(70%)
        self.battery_health = 95 # battery health in percentage
        self.distance__travelled = 0 # distance in km
        self.energy_consumed = 0 # total energy consumtion in kwh
        self.recovered_energy = 0 # energy recovered from regenerative braking in kwh

        # Function to stimulate energy consumption based on speed and driving habits
        def update__energy__usage(self,speed,braking_force):
          self.current__speed = speed
          # simulate power consumption(higher speed = higher consumption
          if speed > 0:
            power_usage = speed * 0.2 + random.uniform(0.5,2) # simulate a rough consumption model
          else:
            power_usage = 0
          self.power_consumption = power_usage
          self.soc -=(power__usage / self.total_capacity) * 100 / 60 # update SOC per minute
          self.energy__consumed += power_usage / 60 #update total energy consumed (per minute)

          # simulate regenerative braking(recovered energy)
          def regen_braking(self, brake_force):
            if brake_force > 0:
              recovered = brake_force * self.regen_efficiency * self.current_speed / 100 #simulate recovery
            else:
              recovered = 0
            self.soc += (recovered / self.total_capacity) * 100/60 # recharge the soc with recovered energy
            return recovered

        # estimate the remaining range based on the current soc and driving habits
        def estimate_range(self):
          avg_consumption = self.energy_consumed / max(self.distance_travelled, 1) # energy consumed per km
          if avg_consumption > 0:
            remaining__range = self.soc / 100 * self.total_capacity / avg_consumption
          else:
            remaining_range = 0
          return remaining_range
        # Basic driving suggestion for efficiency
        def simulate_drive(ev, duration=10):
          speeds = []
          socs = []
          ranges = []
          energies = []

          for minute in range(duration):
            speed = random.uniform(40,120) # random speed for the simulation
            brake_force = random.uniform(0,1) #simulate random braking force

            ev.update_energy_usage(speed, brake_force)
            remaining_range = ev.estimate_range()

            #append data for visualization
            speeds.append(speed)
            socs.append(ev.soc)
            ranges.append(remaining_range)
            energies.append(ev.energy_consumed)

            # display driving suggestions
            print(f"minute {minute+1}: speed={speed:.2f} km/h, SoC={ev.soc:.2f}%, range={remaining_range:.2f} km")
            print(f"Suggestions:{ev.driving_suggestions()}\n")

            time.sleep(0.5)# Simulate real-time delay

        return speeds,socs,ranges,energies

    # Plot the performance evalution graphs
    def plot_performance(speeds,socs,ranges,energies):
      time_axis = np.arrange(1, len(sppeds)+ 1)

      fig, axs = plt.subplots(2, 2, figsize=(12,8))
      fig, axs = plt.subplots(2, 2, figsize=(10,8))

      axs[0,0].plot(time_axis,speed, label="speeds(km/h)")
      axs[0,0].set_title('Speed olver Time')
      axs[0,0].set_xlabel('time(minutes)')
      axs[0,0].set_ylabel('Soc(%)')

      axs[0,1].plot(time_axis,socs,color='green',label="Soc(%)")
      axs[0,1].set_title('State of charge over time')
      axs[0,1].set_xlabel('time(minutes)')
      axs[0,1].set_ylabel('Soc(%)')

      axs[1,0].plot(time_axis,range,color='orange',label="Estimated Range(km)")
      axs[1,0].set_title('Estimated Range over time')
      axs[1,0].set_xlabel('time(minutes)')
      axs[1,0].set_ylabel('Range(km)')

      axs[1,1].plot(time_axis,energies,color='red',label="Energy consumed(kwh)")
      axs[1,1].set_title('Energy consumption over time')
      axs[1,1].set_xlabel('time(minutes)')
      axs[1,1].set_ylabel('Energy(kwh)')

      plt.tight_layout()
      plt.show()

  #Main function run to the EMS system
  if __name__ == "__main__":
    ev=EVParameters() #initialize the EV parameter class

       print("Starting EV energy managementSystem...\n")
       speeds,socs,ranges,energies = simulate_drive(ev, duration=10) # Simulate to 10-minute drive

       print("Simulation complete.Displaying performance data...")
       plot_performance(speeds,socs,ranges,energies) # Plot the performance data
```

```
       File "<tokenize>", line 105
         if__name__ == "__main__":
         ^
     IndentationError: unindent does not match any outer indentation level
```

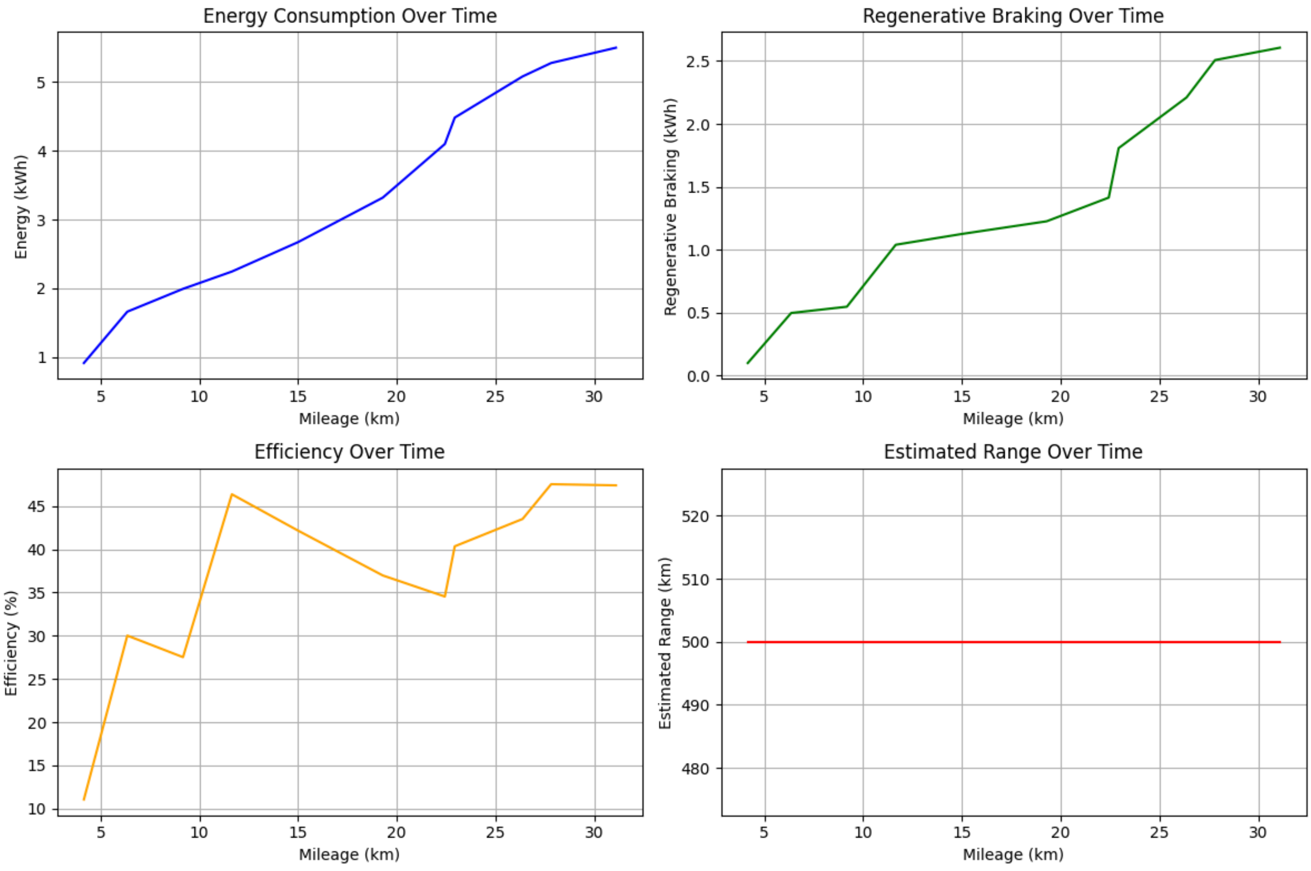
```
import random
import time
import matplotlib.pyplot as plt
import numpy as np

class VehicleData:
    def __init__(self):
        self.battery_health = 100  # Percentage
        self.energy_consumption = 0  # kWh
        self.regenerative_braking = 0  # kWh recovered
        self.mileage = 0  # km driven

    def update_data(self):
        # Simulate energy usage and recovery
        self.energy_consumption += random.uniform(0.1, 1.0)
        self.regenerative_braking += random.uniform(0, 0.5)
        self.mileage += random.uniform(0.5, 5.0)

    def get_data(self):
        return {
            'battery_health': self.battery_health,
            'energy_consumption': self.energy_consumption,
            'regenerative_braking': self.regenerative_braking,
            'mileage': self.mileage
        }

class EnergyManagementSystem:
    def __init__(self, vehicle_data):
        self.vehicle_data = vehicle_data
        self.history = []

    def analyze_data(self):
        data = self.vehicle_data.get_data()
        efficiency = (data['regenerative_braking'] / data['energy_consumption']) * 100 if data['energy_consumption'] > 0 else 0
        range_estimation = self.estimate_range(data['battery_health'])
        self.history.append((data['mileage'], data['energy_consumption'], data['regenerative_braking'], efficiency, range_estimation))
        return {
            'efficiency': efficiency,
            'estimated_range': range_estimation,
            'suggestions': self.generate_suggestions(efficiency)
        }

    def estimate_range(self, battery_health):
        return battery_health * 5  # Assuming 5 km per % battery health

    def generate_suggestions(self, efficiency):
        if efficiency < 20:
            return "Consider smoother driving habits and less acceleration."
        elif efficiency < 50:
            return "You're doing well! Keep maintaining steady speeds."
        else:
            return "Great job! You're optimizing energy well!"

    def plot_performance(self):
        if not self.history:
            print("No data to plot.")
            return

        miles, energy, regen, efficiency, range_est = zip(*self.history)

        plt.figure(figsize=(12, 8))

        plt.subplot(2, 2, 1)
        plt.plot(miles, energy, label='Energy Consumption (kWh)', color='blue')
        plt.title('Energy Consumption Over Time')
        plt.xlabel('Mileage (km)')
        plt.ylabel('Energy (kWh)')
        plt.grid()

        plt.subplot(2, 2, 2)
        plt.plot(miles, regen, label='Regenerative Braking (kWh)', color='green')
        plt.title('Regenerative Braking Over Time')
        plt.xlabel('Mileage (km)')
        plt.ylabel('Regenerative Braking (kWh)')
        plt.grid()

        plt.subplot(2, 2, 3)
        plt.plot(miles, efficiency, label='Efficiency (%)', color='orange')
        plt.title('Efficiency Over Time')
        plt.xlabel('Mileage (km)')
        plt.ylabel('Efficiency (%)')
        plt.grid()

        plt.subplot(2, 2, 4)
        plt.plot(miles, range_est, label='Estimated Range (km)', color='red')
        plt.title('Estimated Range Over Time')
        plt.xlabel('Mileage (km)')
        plt.ylabel('Estimated Range (km)')
        plt.grid()

        plt.tight_layout()
        plt.show()

def main():
    vehicle_data = VehicleData()
    ems = EnergyManagementSystem(vehicle_data)

    try:
        while True:
            vehicle_data.update_data()
            analysis = ems.analyze_data()

            print("\n--- Energy Management System ---")
            print(f"Battery Health: {vehicle_data.get_data()['battery_health']}%")
            print(f"Energy Consumption: {vehicle_data.get_data()['energy_consumption']:.2f} kWh")
            print(f"Regenerative Braking: {vehicle_data.get_data()['regenerative_braking']:.2f} kWh")
            print(f"Mileage: {vehicle_data.get_data()['mileage']:.2f} km")
            print(f"Efficiency: {analysis['efficiency']:.2f}%")
            print(f"Estimated Range: {analysis['estimated_range']:.2f} km")
            print(f"Suggestions: {analysis['suggestions']}")

            if len(ems.history) <= 10:  # Plot every 10 updates
                ems.plot_performance()

            time.sleep(5)  # Simulate time delay for data updates

    except KeyboardInterrupt:
        print("\nTerminating the Energy Management System...")

if __name__ == "__main__":
```

```
main()
```

```
--- Energy Management System ---
Battery Health: 100%
Energy Consumption: 0.91 kWh
Regenerative Braking: 0.10 kWh
Mileage: 4.16 km
Efficiency: 11.05%
Estimated Range: 500.00 km
Suggestions: Consider smoother driving habits and less acceleration.

--- Energy Management System ---
Battery Health: 100%
Energy Consumption: 1.66 kWh
Regenerative Braking: 0.50 kWh
Mileage: 6.35 km
Efficiency: 30.02%
Estimated Range: 500.00 km
Suggestions: You're doing well! Keep maintaining steady speeds.

--- Energy Management System ---
Battery Health: 100%
Energy Consumption: 1.99 kWh
Regenerative Braking: 0.55 kWh
Mileage: 9.17 km
Efficiency: 27.52%
Estimated Range: 500.00 km
Suggestions: You're doing well! Keep maintaining steady speeds.

--- Energy Management System ---
Battery Health: 100%
Energy Consumption: 2.24 kWh
Regenerative Braking: 1.04 kWh
Mileage: 11.65 km
Efficiency: 46.37%
Estimated Range: 500.00 km
Suggestions: You're doing well! Keep maintaining steady speeds.

--- Energy Management System ---
Battery Health: 100%
Energy Consumption: 2.67 kWh
Regenerative Braking: 1.13 kWh
Mileage: 15.00 km
Efficiency: 42.16%
Estimated Range: 500.00 km
Suggestions: You're doing well! Keep maintaining steady speeds.

--- Energy Management System ---
Battery Health: 100%
Energy Consumption: 3.32 kWh
Regenerative Braking: 1.23 kWh
Mileage: 19.29 km
Efficiency: 36.97%
Estimated Range: 500.00 km
Suggestions: You're doing well! Keep maintaining steady speeds.

--- Energy Management System ---
Battery Health: 100%
Energy Consumption: 4.10 kWh
Regenerative Braking: 1.41 kWh
Mileage: 22.44 km
Efficiency: 34.54%
Estimated Range: 500.00 km
Suggestions: You're doing well! Keep maintaining steady speeds.

--- Energy Management System ---
Battery Health: 100%
Energy Consumption: 4.48 kWh
Regenerative Braking: 1.81 kWh
Mileage: 22.94 km
Efficiency: 40.35%
Estimated Range: 500.00 km
Suggestions: You're doing well! Keep maintaining steady speeds.

--- Energy Management System ---
Battery Health: 100%
Energy Consumption: 5.08 kWh
Regenerative Braking: 2.21 kWh
Mileage: 26.37 km
Efficiency: 43.51%
Estimated Range: 500.00 km
Suggestions: You're doing well! Keep maintaining steady speeds.

--- Energy Management System ---
Battery Health: 100%
Energy Consumption: 5.27 kWh
Regenerative Braking: 2.51 kWh
Mileage: 27.82 km
Efficiency: 47.53%
Estimated Range: 500.00 km
Suggestions: You're doing well! Keep maintaining steady speeds.
```

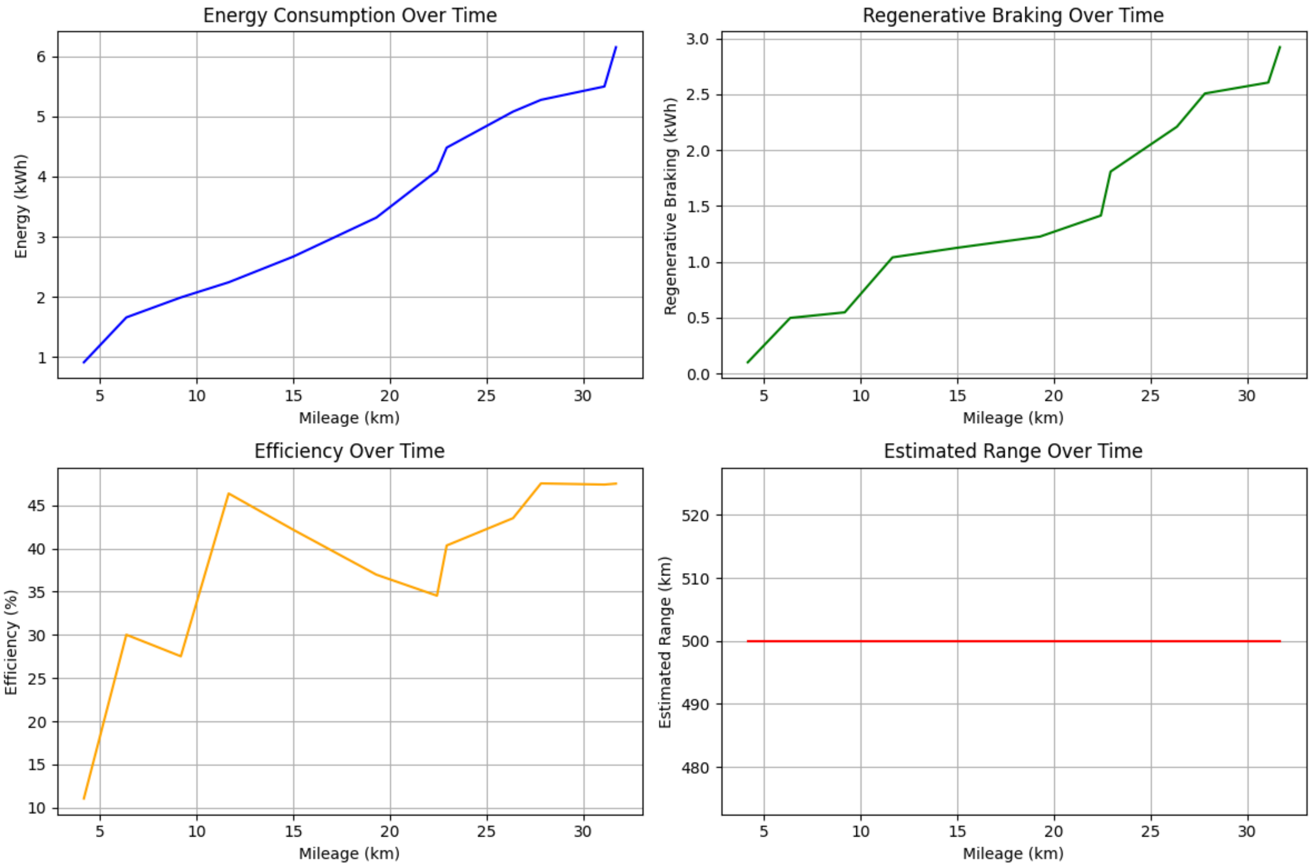


```
--- Energy Management System ---
Battery Health: 100%
Energy Consumption: 5.50 kWh
Regenerative Braking: 2.61 kWh
Mileage: 31.10 km
Efficiency: 47.40%
Estimated Range: 500.00 km
Suggestions: You're doing well! Keep maintaining steady speeds.
```

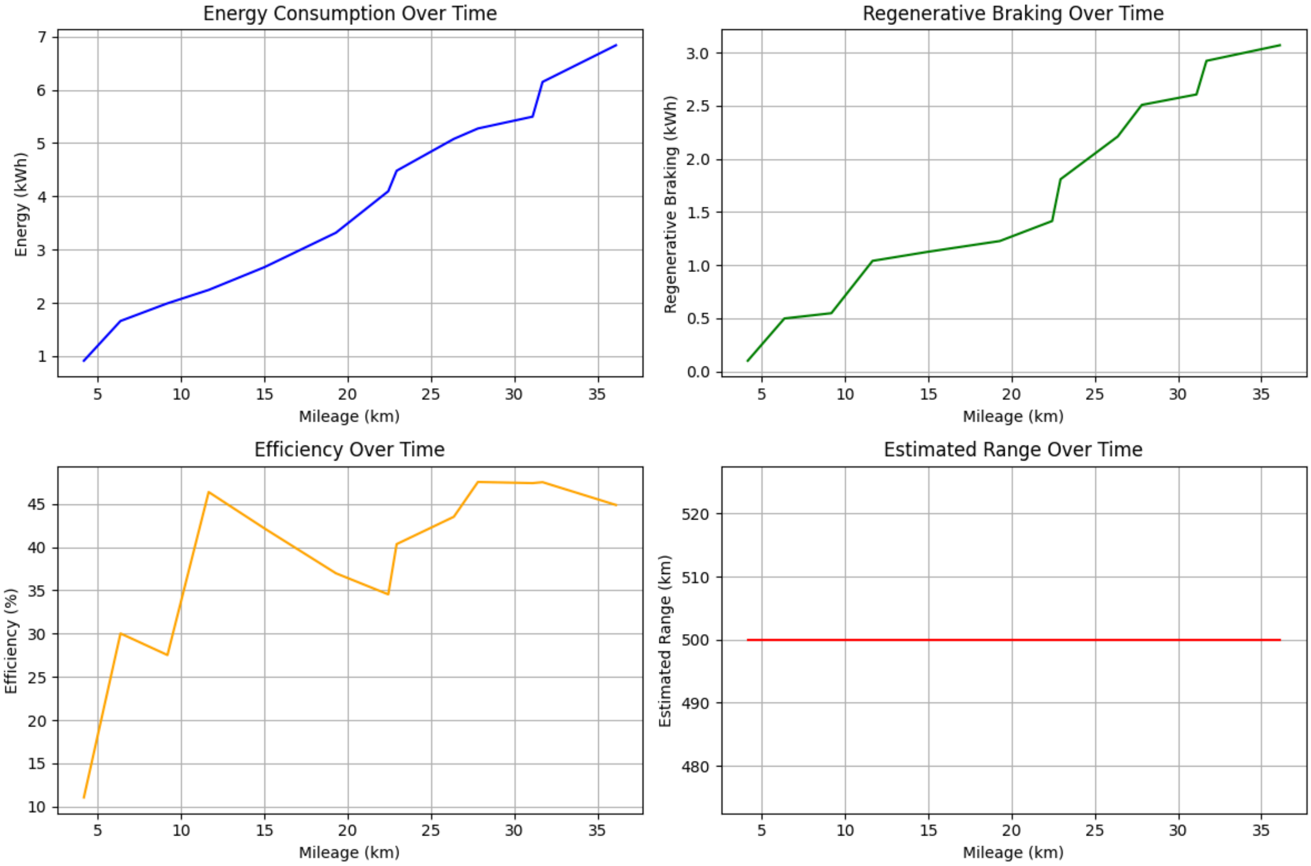


```
--- Energy Management System ---
Battery Health: 100%
Energy Consumption: 6.15 kWh
Regenerative Braking: 2.92 kWh
Mileage: 31.70 km
```

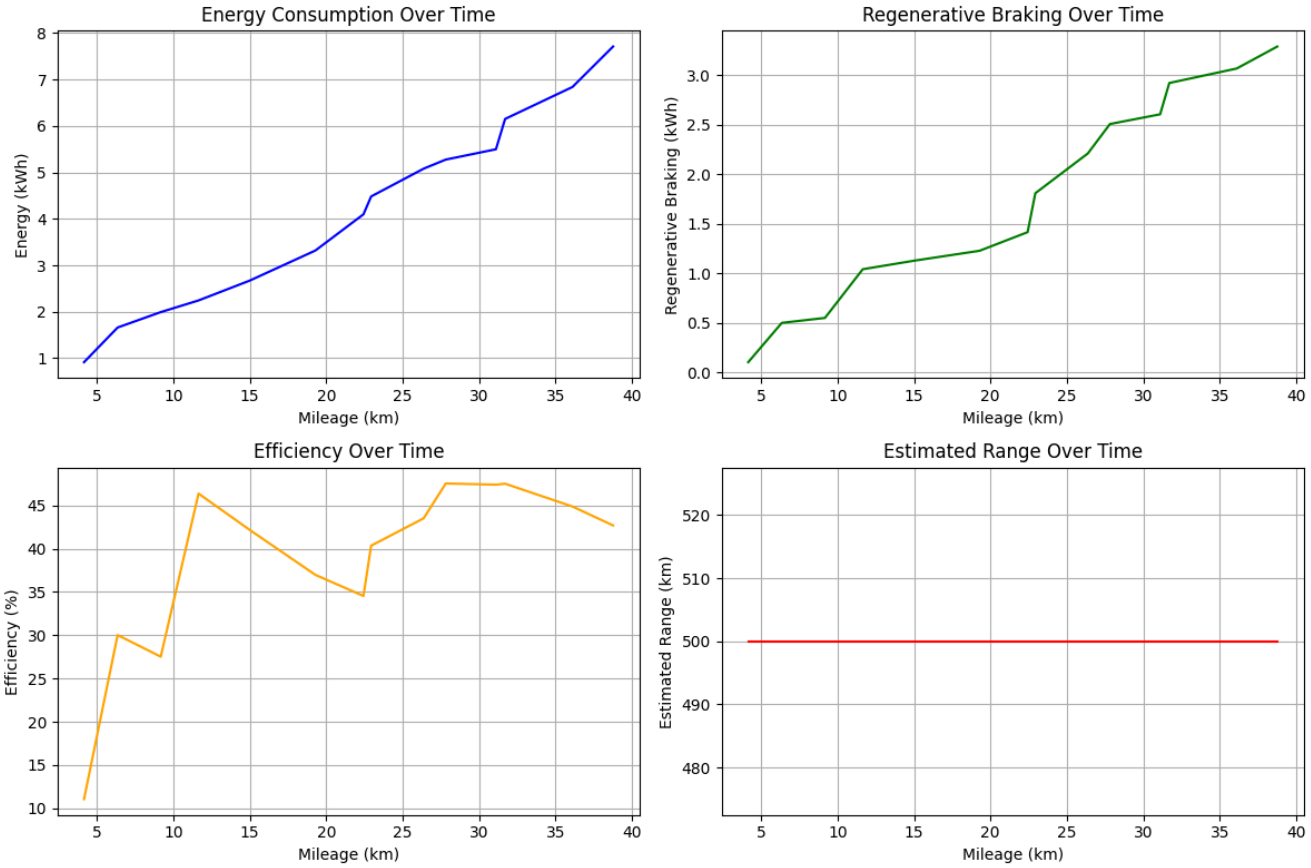
```
Efficiency: 47.51%
Estimated Range: 500.00 km
Suggestions: You're doing well! Keep maintaining steady speeds.
```



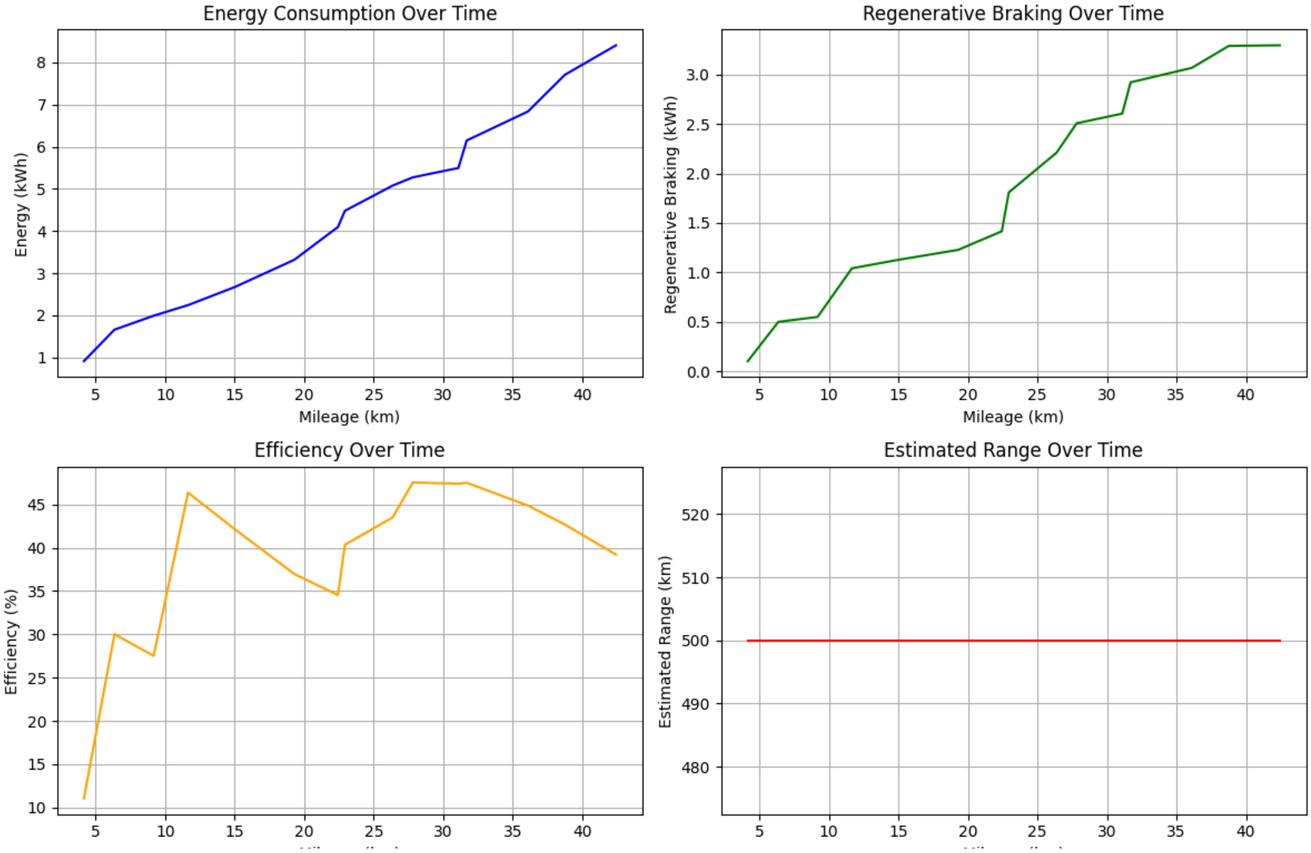
```
--- Energy Management System ---
Battery Health: 100%
Energy Consumption: 6.84 kWh
Regenerative Braking: 3.07 kWh
Mileage: 36.11 km
Efficiency: 44.87%
Estimated Range: 500.00 km
Suggestions: You're doing well! Keep maintaining steady speeds.
```



```
--- Energy Management System ---
Battery Health: 100%
Battery Health: 100%
Energy Consumption: 7.71 kWh
Regenerative Braking: 3.29 kWh
Mileage: 38.77 km
Efficiency: 42.69%
Estimated Range: 500.00 km
Suggestions: You're doing well! Keep maintaining steady speeds.
```



```
--- Energy Management System ---
Battery Health: 100%
Energy Consumption: 8.41 kWh
Regenerative Braking: 3.30 kWh
Mileage: 42.44 km
Efficiency: 39.21%
Estimated Range: 500.00 km
Suggestions: You're doing well! Keep maintaining steady speeds.
```


Energy Consumption Over Time
Energy (kWh)
Mileage (km)
Regenerative Braking Over Time
Regenerative Braking (kWh)
Mileage (km)
Efficiency Over Time
Efficiency (%)
Estimated Range Over Time
Estimated Range (km)